Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D9.3

Frameworks and Approaches for Autonomic Management in Fixed Networks and Ubiquitous Environments

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universidat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, UniBwM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University College London, UCL, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France

Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Frameworks and Approaches for Autonomic Management in Fixed Networks and Ubiquitous Environments

Type: Public

Editors: George Pavlou, Stylianos Georgoulas, Aimilios Chourmouziadis, Antonis Hadjiantonis

E-mail: g.pavlou@ee.ucl.ac.uk, s.georgoulas@ee.ucl.ac.uk, a.chourmouziadis@surrey.ac.uk, a.hadjiantonis@surrey.ac.uk

Authors: Mina Amin, Marinos Charalambides, Gabi Dreo Rodosek, Stylianos Georgoulas, Oscar Fredy Duque Gonzales, Antonis Hadjiantonis, Kin-Hon Ho, Iris Hochstatter, Feng Liu, Cristian Morariu, Krzysztof Nowak, George Pavlou, Javier Rubio, Martín Serrano, Joan Serrat, Szymon Trocha (in alphabetical order)

Doc ID: D9.3-v1.0.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
v0.1	June 23, 2008	Stylianos Georgoulas, Aimilios Chourmouziadis	Integration of all contributions, proofreading, executive summary, introduction, conclusions, references
v0.2	June 25, 2008	Iris Hochstatter	Added CArAM authors, changes to CArAM section
v0.3	June 26, 2008	Joan Serrat	Minor amendments
v0.4	June 27, 2008	Christian Morariu	Minor amendments
v1.0	June 29, 2008	Stylianos Georgoulas, Aimilios Chourmouziadis	Final editing, final version

Legal Notices

Table of Contents

(This page is left blank intentionally.)

1 Executive Summary

Autonomic management is an area that has been receiving significant research interest from both academia and industry in the recent years, since it emerges as an appealing solution to the increasing complexity of managing IT systems. This is because Autonomic Management allows, among others, for self-configuration and self-optimization of the system components. This document provides a detailed view of frameworks and approaches for autonomic management in fixed networks and ubiquitous environments**.** It is the outcome of ongoing research integration and collaboration among partners of EMANICS Network of Excellence and in particular of Work Package 9 (WP9: Autonomic Management).

Having investigated next generation management technologies and approaches to support autonomic management in deliverable D9.2**,** this deliverable builds on top of these technologies and provides specific examples of frameworks and approaches to support autonomic management in fixed networks and ubiquitous environments. These frameworks and approaches were developed in the context of WP9 in order to tackle the following four tasks that were identified as the ones of the greatest importance, interest and challenge in the context of WP9.

- Task 1: Intra and inter-domain autonomic management of fixed networks.
- Task 2: Autonomic management of ubiquitous environments.
- Task 3: Policy based autonomic management.
- Task 4: Peer-to-peer approaches for autonomic network management.

The first task is of great importance since fixed networks carry the vast majority of Internet traffic and provide the backbone over which connectivity and higher level services are built upon. The ability to tackle complexity through autonomic management relieves network administrators and providers from the dubious task of continuous intervention to the management components of their networks. Since connectivity and higher level services need to span in most cases a chain of domains, autonomic management approaches for fixed networks should also take into account inter-domain relations and issues raised.

Ubiquitous networking has been receiving both academic and commercial interest in the recent years. With the proliferation of wireless networks and increasingly networked environments, different approaches have been adopted. Ubiquitous computing is proposed for home networks, while spontaneous approaches to networking focusing on users' interaction and services are also investigated. Different enabling technologies have been considered as the basis for ubiquitous communication. Mobile Ad Hoc Networks (MANETs) offer fast and cheap deployment without the need of existing infrastructure. At the same time, emerging Mesh technologies attempt to combine the benefits of MANETs with the support of wired access points. These networks require different autonomic management paradigms due to their inherent dynamicity and fluidity. This is the focus of Task 2.

Policy based management (PBM) is a generic notion applicable to all types of networks and systems. Policy based approaches to network and systems management are of particular importance because they permit the separation of the rules that govern the behaviour of systems from the functionality provided by that system. The latter means that it is possible to adapt the behaviour of a system without the need to recode

functionality. As a result, changes can be applied without stopping the system. The benefits of policy based management can increase even further, if this is performed in an autonomic fashion. This is the focus of Task 3.

Task 4 relates to peer-to-peer (P2P) approaches for autonomic network management. The distributed nature of the peer-to-peer concepts enhances autonomic network management approaches with the inherent scalability and self-organizing features of peer-to-peer approaches.

As mentioned above, inter-domain relations are of great importance and relevant issues have been identified and addressed in the context of EMANICS. Solutions for these issues have been proposed, both in relation to autonomic management but also in a more generic context. To this end, in this deliverable we will also present work performed in EMANICS to address inter-domain issues.

2 Introduction

The work presented in this deliverable aims to tackle the four tasks presented in the previous section.

Section 3 presents an approach for autonomic management and protection of ubiquitous environments. Special attention is given to security and protection taking into account the characteristics of ubiquitous environments. In such environments, since user devices participate in network management functions, like forwarding and routing, it is essential to integrate monitoring and protection mechanisms that validate the appropriate behaviour of participants and at the same time can cope with selfish or malicious behaviour. Ubiquitous environments are susceptible to having their effective operation compromised by a variety of security attacks. Nodes may misbehave either because they are malicious and deliberately wish to disrupt the network, or because they are selfish and wish to conserve their own limited resources such as power, or for other reasons. Thus, it is necessary to provide these environments with mechanisms of protection that allow them to identify and punish malicious nodes in order to prevent them from causing general disruption. The approach presented in this section aims to address these security/protection-oriented issues.

Section 4 addresses the issue of availability of IT services; the latter plays an essential role for today's IT services. In order to improve service availability, it is necessary for operators to reduce service downtime and deploy mechanisms in order to recover impacted services efficiently in case of outage. This goal can be achieved either by increasing reliability of service components or reducing the service recovery time with well-established recovery plans. In order to alleviate the above mentioned problem, an automated integrated network management framework is presented that optimizes the use of network resources during normal operation and also assists operators to generate recovery plans and actions accordingly, which reduce network re-configuration overhead and service disruptions. The approach presented takes advantage of P2P technologies to achieve some degree of self-organization and to avoid centralization related problems.

In section 5, context management using ontologies and recovery mechanisms for context provisioning in ubiquitous environments is addressed. Context comes from various sources even more so in a ubiquitous environment. Thus, context information has to be gathered and used in different provider domains involving questions of modelling, exchange protocols, trust, accounting, inconsistency, resilience and many

more. In this section, existing context management architectures are investigated and enhanced with components that follow the principles of autonomic management with the aim to facilitate the transition from service agnostic management architectures to service and network resource-awareness.

Section 6 deals with the issue of distributed and autonomic real-time traffic analysis. The constantly increasing data rates available in the core networks of Internet providers and the increasing number of concurrent flows running through the network has become a challenge for traditional centralized IP traffic analysis systems. The main purpose of the work presented in this section is to design distributed and autonomic mechanisms suitable for real-time IP traffic analysis on high-speed network links. An important concern of distributed platforms for IP traffic analysis is the self-organization and self-management of such platforms. The solution proposed allows discovery and adoption of available processing and storage resources for traffic analysis and flow collection in a fully distributed and autonomic manner. Besides that, solutions to automatically anonymize traffic traces, which is especially important for inter-domain traffic analysis and network management scenarios, are investigated.

In section 7, policy based approaches for autonomic intra and inter-domain network management are investigated. As aforementioned, policy based approaches are of particular importance because they allow the separation of the rules that govern the behaviour of systems from the functionality provided by that system. This means that it is possible to adapt the behaviour of a system without the need to recode functionality, and changes can be applied without stopping the system. Policy based techniques and, in particular, policy refinement is intimately linked to the principles of autonomic management. Policy based management can effectively address the issues of self-configuration and self-optimization as two of the key properties of autonomic systems. These are illustrated in realistic case studies presented in this section that demonstrate the applicability of the policy refinement techniques in realistic scenarios.

Section 8 presents inter-domain related issues identified in EMANICS and work that has been performed to address these issues. Finally, section 9 concludes the deliverable, summarizing the main achievements of the work presented in the previous sections.

2.1 Purpose of the Document

The key objective of this deliverable is to present frameworks and approaches for autonomic management in fixed networks and ubiquitous environments. These frameworks and approaches are built on top of the next generation management technologies and approaches to support autonomic management, presented in deliverable D9.2, and are the outcome of ongoing research integration and collaboration among partners of EMANICS Network of Excellence and in particular of WP9.

2.2 Document Outline

This document is organized as follows. After presenting an executive summary for this deliverable, section 2 serves as a general introduction, while it also presents the purpose of this deliverable and its outline. In section 3 we present an approach that aims to address protection and security issues in ubiquitous environments in an autonomic fashion. In section 4 we present an automated framework for optimized operation and failure recovery in IP networks that uses ideas imported from P2P approaches. In section 5 we present the work performed in the area of context management with the aim to enhance existing architectures with the principles of

autonomic management and in section 6 we present solutions for distributed and autonomic real-time traffic analysis that overcomes the deficiencies of traditional centralized IP traffic analysis systems. Section 7 presents extensive work performed in the area of autonomic policy based management, for both intra and inter-domain network management and section 8 gives an overview of inter-domain related issues identified and addressed in the context of EMANICS. Finally, section 9 provides an overview of the work presented in this deliverable.

3 Autonomic Management and Protection of Ubiquitous Environments (AMPUE)

3.1 Introduction

Ubiquitous networking has received both academic and commercial interest. With the proliferation of wireless networks and increasingly networked environments, different approaches have been adopted. Beyond our controlled gadgets, myriads of devices require and expect our interaction in highly networked urban and home environments. Different enabling technologies have been considered as the basis for ubiquitous communication and its management. Mobile Ad Hoc Networks (MANETs) offer fast and cheap deployment without the need of existing infrastructure while emerging Mesh technologies attempt to combine the benefits of MANETs with the support of wired access points. These networks are characterised by their inherent dynamicity, fluidity and the sheer number of users/devices interacting, which takes their management to a degree of complexity that requires more and more human power to maintain it under control. Thus, ubiquitous networking requires management paradigms that hide complexity away from users regardless of whether they use or manage the network. Autonomic systems form part of these new management paradigms. An autonomic system is a system capable of managing itself and adapting to changes in accordance with policies and objectives defined by a human administrator as well as the current state of their surroundings. Such a system can interact with other autonomic systems to perform collaborative tasks in order to achieve or maintain a given goal. Policy based approaches are a first step to provide autonomic management. Policy based management simplifies the complex management tasks of large scale systems, since policies monitor the network and automatically enforce appropriate actions in the system [1].

3.2 Policy Based Management Framework for Ubiquitous Networking in Urban Spaces

The objective is to provide a unified and simplified management solution for networked urban spaces. The nature of these networks sets different requirements, compared to traditional management of fixed ones. More than one manager may have different management objectives. It is essential to detect and resolve conflicts among managers in order to avoid inconsistencies. In addition, users' devices participate in the network but the users require respect for their privacy and preferences. The high mobility environment, in which a user interacts, has an inherent ad hoc element, since they intermingle with users and services on the move. The following sections describe the key aspects of our work, in an effort to design a policy based management framework for networking in urban spaces.

3.2.1 Organisational Model and Multi-manager Paradigm

We apply the organizational model described earlier to a specific case study, i.e. the management of urban spaces under a unified policy based system. The multi-manager paradigm is ideal for the described case study. The assignment of nodes to the role of a Manager Node (MN) needs to be static, in order to ensure that the “eligible entities” are always selected. An “eligible entity” is a public or commercial organization which has some interest in the management of the ubiquitous network. This interest can be either commercial exploitation of the network by providing value added services or regulatory safeguard of the data and functionality of networked devices. The proposed multi-manager paradigm enables the coexistence of more than one “eligible entities” as Manager Nodes (MNs). Each MN can introduce policies in the system to express its high level goals and these policies are interpreted in the management logic of the network. The distribution of the policies among the hypercluster nodes helps on one hand to distribute the load of management and decision making and on the other hand gives localized control to Cluster Heads (CHs) [2]. However, the coexistence of distinct administrative authorities raises the issues of conflict detection and resolution. As it will be discussed later, we incorporate a conflict analysis mechanism in our system to alleviate the problem.

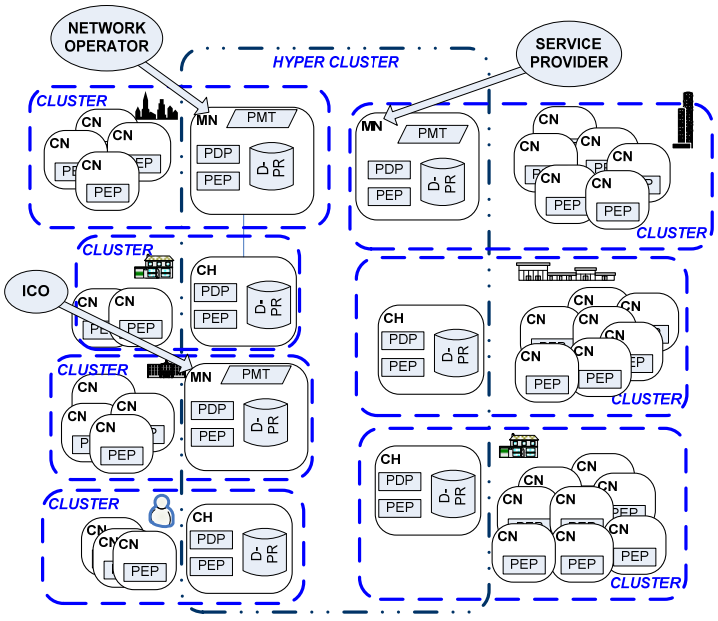


Figure 3-1. Organizational model in an urban space.

For the examined case study of urban space networking, “eligible entities” can be Network Operators (e.g. Mobile Networks Operators), Service Providers (e.g. Multimedia providers), Local Authorities (e.g. Tourism Office), and Data Protection Agencies (e.g. Information Commissioner’s Office - ICO). To demonstrate our ideas, we choose three entities with competing interests in managing the network: a Network Operator, a Service Provider and a Data Protection agency. Figure 3-1 displays the deployment of the proposed organizational model in an urban space. Table 3-1 describes the PBM components each node consists of.

Policy based management can be seen as a way to implement partial autonomy of clusters, by providing them with the management logic (expressed within system policies) and letting clusters decide, based on their preferences and local conditions. At

the top hierarchy level, network managers need only high-level information and do not need to know about the specifics within each cluster.

3.2.2 Policy based Management Architecture

The principles of a policy based management framework have been introduced in [2]. In this work we extend the PBM framework to accommodate the needs of urban space networks. Table 3-1 summarizes the components of PBM systems. DPR is a distributed version of a traditional Policy Repository.

Components		Active for	PBM functions
PMT	Policy Management Tool	MN	introduce, edit
DPR	Distributed Policy Repository	MN, CH	store, distribute
PDP	Policy Decision Point	MN, CH	monitor, decide
PEP	Policy Enforcement Point	MN, CH, CN	enforce, report

Table 3-1. Policy based management components.

The complexity of the environment and the vast numbers of devices make up a challenging environment where the deployment of a policy based system can significantly simplify management tasks and accelerate devices' configuration. There are several policy types necessary in order to effectively manage urban networks:

1. Location-Based Services (LBS) policies.
2. Content delivery policies.
3. Network-wide Preferences policies.
4. Charging policies.
5. Security policies.

We choose a subset of the above in an effort to demonstrate the applicability of PBM through examples applied to the management of networked urban spaces. Location-Based Services (LBS) policies can provide a rich and customizable experience to a mobile user, depending on their physical location as well as their privacy settings. Content delivery policies can control the information that a user receives while at home or on the move. Network-wide Preferences policies can provide users with the recommended settings and the parameterization of their controlled devices.

For simplicity and clarity, we use a restricted notion for policy specification. Policies follow the established event-condition-action (ECA) specification and can be easily adapted to a complete policy language (e.g. Ponder). For the examples of the defined policies, events are omitted since policies are grouped under the same triggering events. A description of the event is provided for better understanding. To complement the design of our PBM architecture, we employ a mechanism for the detection and resolution of policy conflicts. A number of conflicts may arise in the policy specification, like modality and mutual exclusion conflicts, conflicts of duty and multiple manager conflicts. This work focuses on the last type and addresses the inconsistencies that can occur within the adopted multi-manager paradigm.

3.2.3 End-user Privacy Protection

When it comes to managing a network where the networked devices belong to individuals rather than organizations, issues like privacy and data protection should be considered. In European Union, for example, strict legislation by the European Data Protection Supervisor (EDPS, Directive 95/46/EC, http://www.edps.europa.eu) mandates the processing and acquisition of personal data and national authorities have been established to monitor their enforcement (e.g. ICO in UK). Different regulations apply in the US, where a territorial approach is adopted. It is evident that the management of a network consisting of individuals' devices should or is legally obliged to respect the directives regarding the collection and processing of personal data. In order to tackle this issue a two-fold protection mechanism is incorporated in the proposed policy based management framework:

1. ***User-centric control***: Individuals can set their privacy preferences to their controlled networked devices and explicitly restrict access to their personal data, regardless of the network policies.
2. ***Policy based regulation scheme***: The national or regional data protection authority has the ability to introduce appropriate policies to the managed system that will ensure users' personal data are not collected or exploited.

The described case study refers to a trusted ubiquitous environment and we assume that the network is always managed by trusted entities. The requirement is to respect users' preferences and safeguard the unfair use of their personal data; therefore we propose a scheme that prevents manager entities to acquire information against the users' will. The case of non-trusted environments poses the requirement of rigorous security schemes and malicious node detection which are out of the scope of this work. The next section introduces a differentiation between managed objects to accommodate the needs of user-centric control.

3.2.4 Policy Free and Policy Conforming Objects

We establish the definition and differentiation between Policy Free Objects (PFO) and Policy Conforming Objects (PCO) by indicating the benefits and complications imposed to the system. The motivation behind this differentiation is presented here.

Network management can be seen as a set of operations on managed objects in order to achieve effective FCAPS (Fault, Configuration, Accounting, Performance, and Security) management, as defined by ISO. Traditionally, a human network manager can control every managed object (MO) in the system by setting or retrieving values, monitoring the status and reacting to reported events. In other words, a central administrative authority owns and controls the managed network. But, as previously explained, the case of ubiquitous networking in urban spaces is fundamentally different from traditional networks. The individual users are reluctant to entrust the management of their devices to a central authority and demand more control over their owned devices. This contradiction has motivated our idea to differentiate MOs and introduce Policy Free Objects and Policy Conforming Objects.

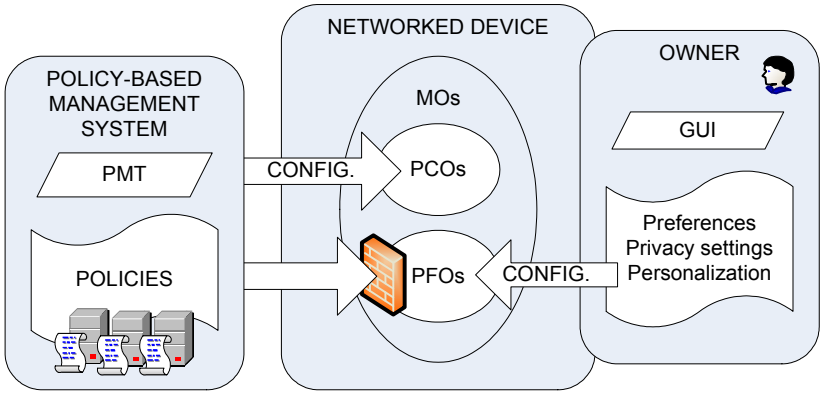


Figure 3-2. Policy free and policy conforming objects.

A policy based management system automates the control of network devices, by enforcing policies over their MOs. We define Policy Free Objects as the MOs of a networked device which are directly controlled by the device's owner and their values and/or status are not influenced by policy decisions. Policy Conforming Objects, similarly to traditional MOs, are controlled by the PBM system, i.e. their values and/or status are influenced by policy decisions. Figure 3-2 presents conceptually the above definitions.

3.3 Protecting Ubiquitous Environments against Packet Forwarding Misbehaviour

The wireless nature and inherent features of mobile ad hoc networks make them vulnerable to a wide variety of attacks by misbehaving nodes. A node may exhibit misbehaviour because it is overloaded, broken, compromised or congested, in addition to intentionally being selfish or malicious [3], [4]. An *overloaded* node lacks the CPU cycles to attend its local and/or network tasks, which leads it to drop packets owing to its inability to process them. A *broken* node has a software or hardware fault that prevents it from performing its network duties properly. A *compromised* node may be victim of an attack that degrades its data forwarding capabilities. A *congested* node receives more packets than the bandwidth available to it allows it to send, its buffer fills and, eventually, it has to drop incoming packets. A *selfish* node is unwilling to use its resources, such as battery life, bandwidth or processing power to forward packets on behalf of other nodes. A *malicious* node drops packets or generates additional packets solely to disrupt the network performance and prevent other nodes from accessing any network services (a denial of service attack). Both selfish and malicious nodes expect, however, other nodes to forward packets on their behalf in spite of their own misbehaviour.

Misbehaviour can be divided into two categories [4]: routing misbehaviour (failure to behave in accordance with a routing protocol) and packet forwarding misbehaviour (failure to correctly forward data packets in accordance with a data transfer protocol). In this work we focus on the latter. Our approach consists of an algorithm that performs two tasks: a) enables packet forwarding misbehaviour detection through the *principle of conservation of flow* [5], and b) enables the accusation of nodes that are consistently detected to be exhibiting packet forwarding misbehaviour. A node that is accused of misbehaviour is denied access to the network by its peers, which ignore any of its transmission attempts. Thus, misbehaving nodes are isolated from the rest of the network. Our scheme is not tightly coupled to any specific routing protocol and,

therefore, can operate regardless of the routing strategy adopted. Our criterion for judging the detection of a node is the estimated percentage of packets dropped, which is compared against a pre-established misbehaviour threshold. Any node dropping packets in excess of this threshold is considered to be a misbehaving node while those dropping packets below the threshold are considered to be correctly behaving.

Our scheme detects and accuses misbehaving nodes (whether selfish, malicious or otherwise) capable of launching two known attacks: the simplest of them is the black hole attack. In this attack a misbehaving node drops all the packets that it receives instead of normally forwarding them. A variation on this is a grey hole attack, in which nodes either drop packets selectively (e.g. dropping all UDP packets while forwarding TCP packets) or drop packets in a statistical manner (e.g. dropping 50% of the packets or dropping them with a probabilistic distribution). Both types of grey hole attacks seek to disrupt the network without being detected by the security measures in place.

3.3.1 Terminology

We use the term *neighbour* to refer to a node that is within the direct wireless transmission range of another node. From this, it follows that both nodes are able to establish a reliable bidirectional communication. Likewise, the term *neighbourhood* refers to all nodes that are neighbours of a particular node. A node is not a neighbour of itself and, therefore, a node does not belong to its own neighbourhood.

We use the term *detection* to mean that our algorithm has identified that a node appears to be misbehaving. Detection is based on a *single* check of the node's behaviour. An *accusation*, on the other hand, occurs when a node reports another node as misbehaving. It is our view that an accusation should be based on more than a single positive detection to increase confidence in the assessment.

3.3.2 Detecting and Accusing Misbehaving Nodes

Our work provides a novel methodology to secure the data forwarding functionality in mobile ad hoc networks. We propose an approach that takes advantage of the principle of flow conservation in a network. This states that all bytes/packets sent to a node, and not destined for that node, are expected to exit the node.

The core parts of our algorithm are detailed in the pseudocode shown in Figure 3-3. A node v_i maintains a table with two metrics T_{ij} and R_{ij} (Figure 3-3a), which contain an entry for each node v_j to which v_i has respectively transmitted packets to or received packets from. Node v_i increments T_{ij} on successful transmission of a packet to v_j , for v_j to forward to another node, and increments R_{ij} on successful receipt of a packet forwarded by v_j that did not originate at v_j.

All nodes in the network continuously monitor their neighbours and update the list of those they have heard of recently (Figure 3-3b). If the ID of an overheard node is not included in the table of overheard nodes, then a new entry is created. Otherwise, the existing entry is updated with a timestamp corresponding to the time the node was last overheard. Upon the creation of a new entry, a node schedules a task/event to check the behaviour of the node whose ID has been saved in the new entry. Nodes randomly select a period of time between T_{min} and T_{max} to schedule the behaviour checking task. This random selection seeks to reduce the possibility of two or more nodes starting a behaviour check on the same node at the same time, wasting network bandwidth, battery energy and other network resources.

a. MONITORING
if node v_i successfully sends a packet to node v_j
. increase T_{ij}
endif
if node v_i receives a packet successfully forwarded by node v_j
. increase R_{ij}
endif

c. INITIATE BEHAVIOR CHECK
if in node v_i an event to check node v_j's behavior is triggered
. send a metrics request packet (MREQ) with node v_j's ID
. schedule another event to check v_j's behavior again at $t+T_{max}$
endif

e. REPLY HANDLING
if a request was sent out
. while there are more replies to be received for node v_j
. . receive reply
. . acknowledge reply reception (send MACK)
. . add received metrics to totals
. endwhile
.
. if $(1-\alpha_{threshold}) \sum_{\forall i | v_i \in U_j} R_{ij} \leq \sum_{\forall i | v_i \in U_j} T_{ij}$
. . node v_j is misbehaving (detection)
. . send a detection alert packet (DAP) with node v_j's ID
. else
. . node v_j is not misbehaving (non-detection)
. endif
endif

g. ACCUSATION HANDLING
if node v_i receives an accusation packet for node v_j
. if node v_i has node v_j in its accusation table
. . ignore accusation packet
. else
. . add node v_j to v_i's accusation table
. . rebroadcast accusation packet (AP)
. endif
endif

b. BEHAVIOR CHECK SCHEDULING
if node v_i overhears a node $v_j \in U_k$
. if node v_j is not in v_i's table of overheard nodes
. . add node v_j to v_i's table of overheard nodes
. . schedule an event to check v_j's behavior
. else
. . update last time node v_j was heard
. endif
endif

d. REQUEST HANDLING
if node v_i receives a metrics request for node v_j
. if node v_i has node v_j in its table of overheard nodes
. . rebroadcast metrics request packet (MREQ)
. . reschedule any event to check v_j's behavior
. . if node v_i has metrics for node v_j
. . . send a metrics reply (MREP) back to the requesting node
. . endif
. else
. . ignore request
. endif
endif

f. DETECTION ALERT HANDLING
if node v_i receives a detection alert for node v_j
. if node v_i has node v_j in its table of overheard nodes
. . rebroadcast detection alert packet (DAP)
. . if max_num_of_detections for node v_j has been reached
. . . broadcast na accusation packet (AP) with node v_j's ID
. . endif
. else
. . ignore detection alert
. endif
endif

h. PUNISHING ACCUSED NODES
if node v_i receives a packet from node v_j
. if node v_j is in node v_i's accusation table
. . ignore packet
. else
. . handle and process the packet
. endif
endif

Figure 3-3. Our algorithm pseudocode.

When a scheduled task is triggered in node v_i to check v_j's behaviour (Figure 3-3c), node v_i broadcasts a *metrics request packet (MREQ)* with TTL = 1 in the IP header. An MREQ includes the ID of the node emitting the request (SRC_ID*)*, the ID of the node whose behaviour is to be checked (CHK_ID*)*, an MREQ_ID and a timestamp indicating the time at which the task was triggered. The MREQ_ID is used in the same way as in some routing protocols which base their route discovery phase on broadcasting. If a node sees an MREQ that has the same MREQ_ID and SRC_ID of a packet seen before, the MREQ is dropped. This technique prevents flooding packets from traversing a zone of the network more than once. The timestamp, on the other hand, is used to resolve conflicts when two nodes start a behaviour check on the same node at almost the same time. In such cases, nodes can see which of the packets corresponds to the earlier triggered task and disregard the others. Nevertheless, two or more unsynchronized nodes performing a behaviour check on the same node will generate different timestamps. In this case nodes receiving the MREQ packets will select the packet with the earliest timestamp and will reply accordingly. Thus, our approach does not require accurate synchronization of the nodes' clocks. Finally, after the MREQ packet is broadcasted, a task is scheduled to be triggered at a period of time T_{max}

(maximum elapsed period of time without checking an active node's behaviour) later. This means it is highly unlikely that the same node will originate two successive checks of another node, and gives other nodes a chance to perform the behaviour check.

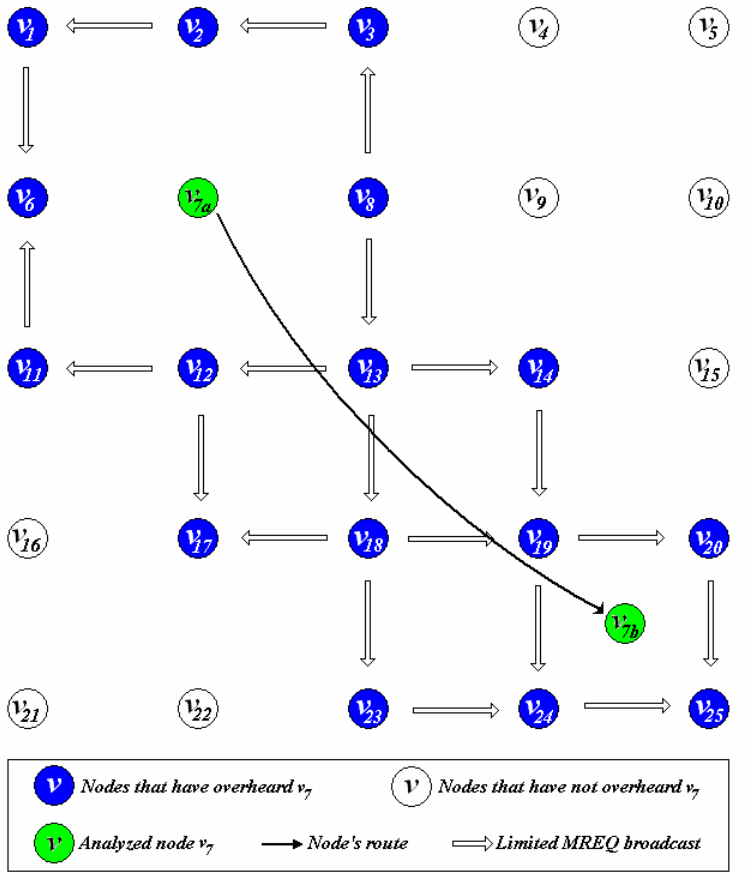


Figure 3-4. Example of limited broadcast to track down nodes that have overheard node v7.

The handling of requests (Figure 3-3d) illustrates the heart of our limited broadcast algorithm. When a node receives an MREQ it first checks if the CHK_ID is in its table of overheard nodes; if it is not the node ignores the MREQ and discards the check. However, if the CHK_ID appears in its table then it rebroadcasts the MREQ with TTL = 1 in the IP header. Setting the TTL to one allows our algorithm to control how far the broadcast of the MREQ is to go, instead of leaving this task to the IP protocol. Thus, every MREQ travels only one hop at a time, and is then analyzed and rebroadcasted if the protocol so determines. By following this algorithm, our protocol is capable of tracking down nodes that have been in contact with the checked node, as illustrated in Figure 3-4. We assume transmissions can be overheard by vertically, horizontally and diagonally adjacent nodes. In Figure 3-4, node v_7 is first in position *a* where it can be overheard by nodes v_1, v_2, v_3, v_6, v_8, v_{11}, v_{12} and v_{13}. Each of these nodes makes an entry in their table of overheard nodes when v_7 first transmits and each of them schedules a task to check its behaviour. At some point in time, v_7 decides to move following the path depicted in Figure 3-4 coming in contact with nodes v_{14}, v_{17}, v_{18}, v_{19}, v_{20}, v_{23}, v_{24} and v_{25}. It finally stops in position *b*. The scheduled behaviour check initiation task (see Figure 3-3c) in v_8 is the first to be triggered and the limited broadcast commences. All nodes that have overheard node v_7 rebroadcast the MREQ, whereas nodes such as v_4, v_9 and v_{15} also receive the MREQ but ignore it because they have not overheard node v_7.

It may also happen that node v_7 stops transmitting and receiving packets before it moves to a different network area. Then, after moving, v_7 may become active again forming a new neighbourhood. In this case the old and new neighbourhoods are not connected by nodes that have overheard v_7 and, therefore, a limited broadcast triggered in one neighbourhood will not reach the other. In spite of this, our algorithm still works properly because two independent behaviour checks will be performed on v_7: one at its old neighbourhood and another at its new neighbourhood. The outcome of each of these behaviour checks depends on the behaviour exhibited by v_7 at each neighbourhood.

Once a node has decided whether to continue or not broadcasting an MREQ, it reschedules any pending task to check the behaviour of the checked node specified in the CHK_ID field of the MREQ. The new behaviour checking task is scheduled in the same way as when a new entry is made in the table of overheard nodes, i.e. a period of time is randomly selected between T_{min} and T_{max}.

The last task a node performs when it receives a MREQ is to check if it has any metrics (R_{ij} or T_{ij}) relating to the node being checked. If any of the metrics has a value other than zero the node returns a metrics reply packet (MREP) (Figure 3-3d) containing the metrics, but if the value of both metrics is zero then the node does not send back any response. In our scheme a metrics reply packet is returned to the node that originated the MREQ following the reverse of the MREQ's path. This requires nodes to set a backward pointer when they receive an MREQ. This approach avoids the high overhead produced by reactive routing protocols when they perform route discovery for many MREP packets.

Reply handling is executed in the node that initiated the MREQ. This node, v_8 in Figure 3-4, waits for a period of time in order to give to all nodes, with metrics about the checked node, the opportunity of replying. When the time expires, the node checks the behaviour of the analyzed node by verifying that equation in Figure 3-3e holds. If it does not, this is considered a detection and a detection alert packet (DAP) containing the detected node ID is broadcast with TTL = 1 in its IP header. Detection alert packets are broadcasted in the same way as MREQs, i.e. they follow our limited broadcast algorithm (Figure 3-3f).

Due to the nature of our algorithm, nodes are not perfectly synchronized with each other. An MREQ will reach close nodes faster than nodes placed a few hops away. The last nodes to receive the MREQ have enough time to send or receive some extra packets to and from the analyzed node, thus unbalancing the values of the T_{ij} and R_{ij} metrics. This discrepancy, in which some packets may have been sent to the node being analyzed but not yet forwarded by it, is accommodated by $\alpha_{threshold}$ in equation in Figure 3-3e.

A problem that has been detected in our simulations has its roots in the dynamic nature of MANETs. Nodes receiving a MREQ with non-zero metrics for the checked node send back their reply. However, such replies sometimes get lost due to collisions, noise in the wireless channel or link/path breakages due to the mobility of the nodes. If the value of the metrics contained in the lost reply is small compared to the total obtained after adding up the replies that do not get lost, $\alpha_{threshold}$ can accommodate them. Unfortunately, this is not always the case and some of those replies that get lost contain key information for the calculations and the checked node is then falsely detected as misbehaving. This is one of the reasons why an accusation should not be made based on a single detection. Using a single detection to accuse a node is not sufficient since

such an approach may lead to false accusations against correctly behaving nodes. Our scheme, in which multiple detections by different nodes are necessary to accuse a node, is fairer to well-behaved nodes, while keeping a high probability of correctly accusing misbehaving nodes. Additionally, to circumvent the lost replies problem we propose an optional module to our algorithm. A node receiving an MREP as it is forwarded towards its destination (i.e. the node performing the behaviour checking task) will also send a metrics acknowledgement packet (MACK). Thus, nodes sending/forwarding an MREP wait for an MACK from the next hop in the route. If the confirmation does not arrive then they retransmit the MREP. The process is repeated up to MAX_{Retx} retransmission retries before giving up. The results obtained in our simulations have demonstrated that this technique can significantly improve the results in MANETs with a high degree of mobility. Simulations have also shown that the most significant improvement can be seen when comparing the results for MAX_{Retx}=0 (without retransmitting any reply) and MAX_{Retx}=1. Subsequent increases to MAX_{Retx} improve the results further but not significantly.

The handling of detection alerts (Figure 3-3f) is also determined by our limited broadcast algorithm. This means that the information to accuse a node of misbehaviour is collected locally rather than globally. When a node receives a detection alert packet (DAP) it first checks if the reported node ID contained in the received packet is present in its table of overheard nodes; if it is not, the node stops broadcasting the DAP. However, if the ID appears in its table then it rebroadcasts the DAP with TTL = 1 in the IP header. Thus, nodes can control how far the DAP broadcast is to go in the same way they do with an MREQ. For instance, assuming that v_8 in Figure 3-4 detects that v_7 is misbehaving, the DAP follows the same path as that depicted in Figure 3-4 for a MREQ packet. Although this approach prevents nodes from generating excessive network overhead, it may also permit malicious nodes constantly changing their geographical position in a clever manner (without going back to previously visited areas) to avoid being accused. After a node has decided whether to continue broadcasting a DAP or not, it checks if an entry for the reported node ID has been already created in its detections table. If it has not, a new entry is created with its field number of detections equal to one. If the entry is already present, its number of detections is increased and then compared against a *detections threshold*. When the number of detections reaches the detections threshold in less than a predefined period of time $T_{threshold}$, there is enough evidence to accuse the reported node of misbehaviour. Therefore, an accusation packet (AP) is broadcasted in a network wide fashion so that access is denied to the reported node all over the network.

Nodes that receive an accusation packet (Figure 3-3g) examine their accusation tables to see whether the reported node has been accused previously. When an AP with a newly accused node is received, a new entry is created in the accusation table and the broadcast of the AP continues. Otherwise, the packet is ignored and dropped to prevent unnecessary network traffic. Finally, all nodes in the MANET are responsible to ensure that packets coming from an accused node (a node present in their accusation table) are immediately dropped (Figure 3-3h). This approach denies misbehaving nodes any chance to have their packets transmitted in the network as well as to participate in route discovery, thus preventing them from causing further disruption in the communication process.

3.4 Conclusions

We have presented a novel policy based framework for the management of the increasingly networked urban spaces. The framework offers each user the opportunity to set their individual privacy settings and preferences, regardless of the network policies, while regulatory bodies can monitor the acquisition of user data and investigate unfair exploitation. By adopting a multi-manager scheme we allow more entities to offer different services to the users, without violating their privacy concerns and preferences. The framework integrates an automated conflict detection and resolution mechanism that prevents policy inconsistencies among different managers.

The self-regulating nature of MANETs requires that they are able to monitor the behaviour of the network. Limited resources mean that there is an incentive for nodes to misbehave by not correctly forwarding packets (selfish nodes); nodes may also misbehave for other reasons.

We have presented an algorithm that is capable of detecting and accusing nodes that exhibit packet forwarding misbehaviour. The algorithm does not require high density networks in which many nodes can overhear each others' received and transmitted packets, but instead uses statistics accumulated by each node as it transmits to and receives data from its neighbours. Also, our algorithm does not interfere with the routing protocol which allows for its use regardless of the routing strategy employed.

Selecting an acceptable or tolerable level of misbehaviour in a network is a policy decision that enables us to choose an adequate misbehaviour threshold $\alpha_{threshold}$. However, this threshold also depends on dynamic network parameters such as node density and network area. Therefore finding a way to calculate an optimal misbehaviour threshold in a dynamic manner is of great importance, especially in autonomic environments where the network should automatically adjust its parameters so as to adapt itself to changes in its surroundings. In this respect our future research will focus on the study of methods to collect and synthesize network context information, and techniques to calculate an adaptive misbehaviour threshold using such information.

4 Automated Service Management using Emerging Technologies (ASMET)

4.1 Introduction

Availability of networks is becoming more essential in today's networked world. Whereas network faults are inevitable facts, an efficient fault management process is decisive to decrease the fault resolution time and to increase the availability of the network. In case of failure, an efficient fault recovery process is expected to find fault resolutions quickly and recover the impacted network in a timely manner. Hence, recognizing the importance of minimum system downtime to maintain compliance with accepted Service Level Agreements (SLAs), we propose a dual stage fault recovery process. The first stage is a short-term system reaction to minimize the immediate effects of a fault. In order to allow for quick system response, this first reaction is pre-planned in sets of recovery policies able to anticipate faults. The second stage is the long-term recovery plan and aims to discover the recovery knowledge and plan the recovery process.

The presented architecture involves traffic engineering (TE), policy based management

and artificial intelligence approaches. Our research focuses on providing a comprehensive framework to facilitate an automated fault recovery process in large-scale networks. The motivations of the architecture are to:

1. Provide short-term recovery plans through fast recovery solutions;
2. Provide long-term recovery plans through an efficient recovery knowledge discovery approach;
3. Automate recovery planning for the long-term recovery process.

To demonstrate the applicability of our approach, we choose a scenario related to inter-domain traffic engineering and examine a case study of automated recovery from an egress point (EP) failure. Since inter-domain links are the most common bottlenecks in the Internet [6], an efficient plan for fault recovery is necessary. We focus our interest on planning the recovery from border router faults to maintain optimized configuration of outbound inter-domain traffic. Once a border router (EP) fails, we follow a dual stage recovery process that will be detailed in section 4.4

The first stage is to switch affected traffic flows to another EP, while at the same time optimizing the EP selection (border router selection) for all outbound inter-domain traffic of a domain. To achieve a quick reaction that would minimize disruption, we execute in advance an outbound TE algorithm and store short-term recovery plans in a repository. The algorithm is designed with the goal of inter-domain link load-balancing and creates configuration sets for each case of the EP failures. In addition, the algorithm creates the initial configuration set for normal operation which is used until a failure is detected. Having responded to the border router failure and minimizing short-term disruption, the second recovery stage begins to discover a long-term solution and recover from the failure. Once notified about the failure, the recovery system relies on proposed policies and actions to decide an appropriate recovery plan. More specifically, the planning subsystem receives high-level directives and actions as input, and then combines the input with received state information from the monitoring system in order to generate the appropriate sequence of actions for remedying the failed EP and recovering normal network operation.

4.2 System Architecture

Our proposed system contains three subsystems: Automated Recovery Planning (ARP), Knowledge Discovery (KD) and Policy Based Management (PBM), see Figure 4-1. The planning subsystem obtains status reports from the monitoring component in order to carry out analyzing and executing recovery plans for the managed system with the support of the other subsystems that provide appropriate actions and policies respectively. The introduction of each subsystem is presented in the following subsections.

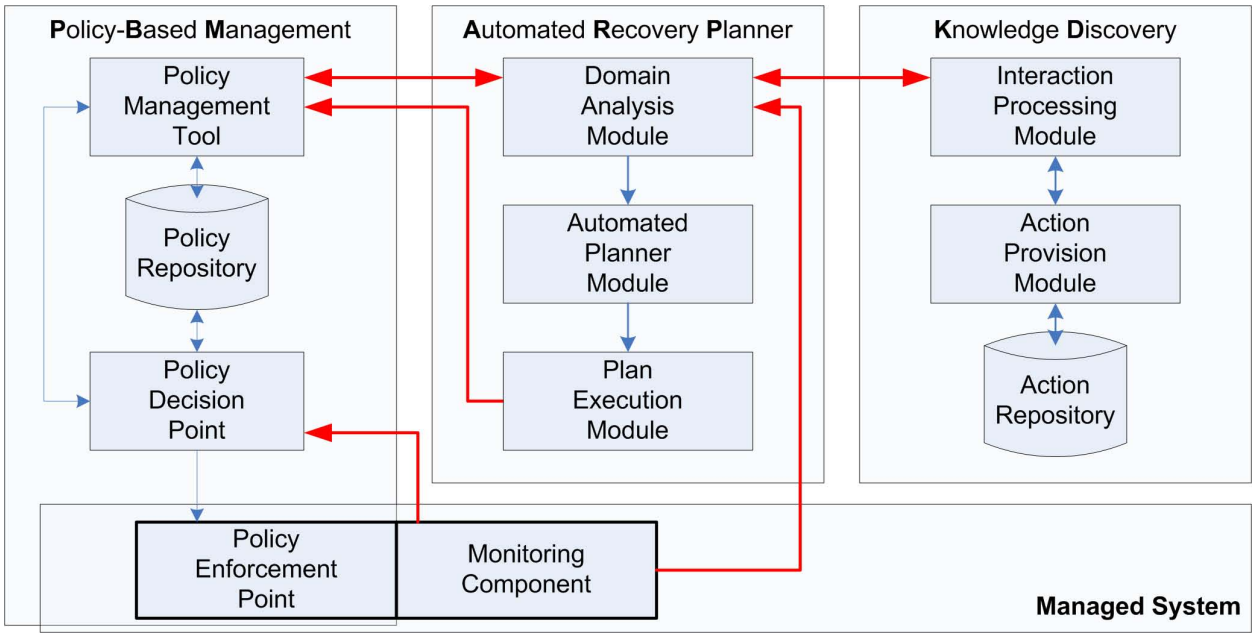


Figure 4-1.Overview of system architecture.

Automated Recovery Planning (ARP): This central subsystem provides fault recovery plans for the managed system. Its tasks involve analyzing the status report from the monitoring component, collecting relevant information from the other subsystems for the recovery activities, producing relevant recovery plans and provisioning for plan executions on the managed system. In Figure 4-1, the three modules responsible for these tasks include:

Domain Analysis (DA): This central module connects to *KD*, PBM subsystems and to the monitoring component for aggregating the relevant information as plan knowledge to facilitate the operation of the automated planner module. Such plan knowledge includes the description of the current system state, e.g. which components are impacted by the failures and which components are still operational etc. Depending on the impacted components, it contacts *KD* for a set of relevant recovery measurements attached with meta-information. The meta-information describes under which situations a particular solution step could be applied and what consequences could be expected from it, i.e. how a particular solution step is going to affect the current system state. *DA* also contacts *PBM* to retrieve recovery policies.

Automated Planner (AP): This module derives recovery plans based on plan domain descriptions provided by *DA*. It contains an automated planning algorithm, which correlates relevant recovery knowledge and produces recovery plans by reasoning. Note that the selected planning algorithm is domain-independent and generic. The rationale to use domain knowledge as input is to accelerate the planning process and to increase the efficiency of the planning algorithm.

Plan Execution (PE): This module makes provision for the recovery plan executions. Plans generated by *AP* are described in a specific plan description language, therefore, to enable the plan executions, it is necessary to convert the format of the plan and map the recovery plan into executable actions. Since *PE* relies on *PBM* for plan execution, it converts the generated plan into a PBM-compatible format and sends the plan to *PBM,* where the mapping and execution of the plan take place. The results will be observed by the monitoring component and sent back to *DA* to see if further recovery measurements are necessary.

Knowledge Discovery: Case-Based Reasoning (CBR) [7] resolves a problem by searching for similar problems and reasoning on their solutions found in the past. The reasoning capability of a conventional CBR system is restricted by a local case database. A distributed CBR system takes advantage of computation power and problem-solving knowledge at various sites, where independent CBR engines can operate in parallel, thus improving the efficiency of resolving complicated problems and the performance of managing huge case databases. This system exploits the integrated framework of Peer-to-Peer (P2P) and CBR [8], which involve retrieving problems and inferring solutions, respectively.

This subsystem acts as a distributed CBR system to provide actions for recovery plans. Its tasks involve communicating with various knowledge sources (e.g. the Internet, the operators), dealing with various requests from and responses to other modules, inferring proper actions corresponding to the failed status of the managed system and managing a case database (e.g. failure cases and actions). These tasks belong to three modules, shown in Figure 4-1.

Action Repository (AR): This module involves storing cases and maintaining the case database. It regularly checks the usage of cases and the similarity of cases in order to deactivate obsolete cases or consolidate cases. Maintaining the case database is essential since the CBR system tends to be lumbering and inefficient with a large number of cases.

Action Provision: This module serves as an independent CBR engine which takes requests and the case database as input to provide actions. Its main tasks include: a) retrieving similar cases from *AR*, b) reasoning on these cases to figure out the most promising case, and c) updating the case database. To improve *AR*, the module regularly updates new cases from various sources including adapting instructions from operators to solutions, or updating cases by learning problems from monitoring systems, or extracting cases from the response of peers.

Interaction Processing: This module is responsible for establishing and maintaining a P2P network. Its main tasks are to interact with various sources including operators, surveillance systems or other peers for information exchange, resource search and lookup. Particularly, it deals with requests from peers forwarded to other peers, instructions and updates from the operators processed to update *AR*.

Policy based Management: *PBM* simplifies the complex management tasks of large scale systems, since high-level policies can be automatically enforced as appropriate network management [1]. In general, policies are defined as Event-Condition-Action (ECA) clauses, where on event(s) E, if condition(s) C is true, then action(s) A is executed. The components of *PBM* are shown in Figure 4-1 along with their interactions with *ARP* and the managed system. The four functional elements, as defined by IETF, are described below:

Policy Management Tool (PMT): PMT is the interface between the operators and *PBM*. It allows the introduction and editing of policies and also provides notification about critical events requiring a manager's attention. Using this interface, the operators carry out the specification of operational policies by selecting the appropriate policies from the supported policy types and selecting the required parameters. We extend *PMT*'s functionality by interfacing it with *ARP*. *PMT* provides relevant policies to *ARP* to assist the planning process.

Policy Repository (PR): *PR* encapsulates the management logic to be enforced on all networked entities, as expressed in policies. It is the central point where policies are stored by managers using *PMT* and can be subsequently retrieved either by the Policy Decision Point (PDP) or by *PMT*.

Policy Decision Point (PDP): *PDP* is responsible for evaluating policy conditions and deciding when and where policy actions need to be enforced. Once relevant policies have been retrieved from *PR*, they are interpreted and *PDP*, in turn, provisions any decisions or actions to the controlled Policy Enforcement Point (PEP). *PDP* receives input from the monitoring component of *PEP* to form a closed control loop.

Policy Enforcement Point (PEP): PEP enforces policy decisions, as instructed by *PDP*. Within our framework, *PEP* is enhanced by the *monitoring component* that reports local information to assist decision making.

The *monitoring component* is integrated to provide feedback to both *PBM* and *ARP* for decision making. It is not formally part of our architecture since fault detection and diagnosis are out of the scope of this work. Critical events such as failures are reported to both subsystems in order to initiate the dual stage recovery process.

4.3 Methodology

In this section, we give detailed views on the methodologies the subsystems apply. The discussion focuses not only on individual subsystems, but also tries to provide an overview on the interactions between them.

4.3.1 PBM and ARP Collaboration

As networks become more and more complex, unavoidably faults occur more frequently. It is evident that frameworks with automated recovery capabilities can significantly expedite and simplify management tasks. Within our framework, policies work in two layers to express on one hand, high-level business objectives and on the other hand, low-level configuration policies to anticipate faults. Policies can encapsulate the overall management logic at two functional layers: the business layer and the device layer. Using sophisticated algorithms, we translate high-level policies defined at the business layer to low-level configuration policies at the device layer. At the same time, the *PBM* interacts with *ARP* to provide policies and constraints to be used in the planning procedure.

By executing an outbound TE algorithm in advance, we create different policy sets that express those high-level goals in normal operation and, in addition, anticipate possible system failures. These low level policies constitute short-term plans for the EP selection aiming at the recovery of outbound inter-domain traffic when an EP fails. The created recovery policy sets are stored in the Policy Repository and the policy set for normal operation is enforced. An outbound inter-Autonomous System (AS) TE algorithm that is able to optimize the EP selection (border router selection) for outbound inter-domain traffic of a domain has been developed in [9]. The algorithm is designed with the goal of inter-domain link load balancing. The output is $|L|+1$ ($|L|$ number of border routers) sets of EP selection configurations, one set for normal operation (i.e. no failure) that can be called PrimaryEgressPoint selection and one set for each case of EP failures that can be called BackupEgressPoint selections. In other words, PrimaryEgressPoint selection determines EP selection under no inter-domain link failure and BackupEgressPoint selection determines EP selection under Failure States (FSs). A common implementation of EP selection is by adjusting the local-preference value in the Border

Gateway Protocol (BGP) route attribute [10]. According to the BGP route selection process, local-preference has the highest priority and its value indicates the preference of the route. The higher the local-preference, the more preferred is the route. The algorithm takes inter-domain connectivity, BGP routing information, inter-domain traffic matrix as inputs and then deterministically calculates the EP selection through local search heuristics. The interested reader can refer to [9] for more details.

High-level policies are used to express goals of inter-domain traffic management, e.g. "*optimize EP selection to achieve inter-domain load balancing*", or "*optimize EP selection avoiding routing traffic to destination prefix p1 using domain D2*". For security reasons we may wish to avoid routing the inter-domain traffic flows towards some particular destinations through some specific domains. The benefit of policy based management is the ability to use different policies to achieve different goals, e.g. minimize peering cost. Furthermore, high-level or recovery policies could also be used by *ARP* in order to find the recovery plans which are in compliance with the management goals. It is possible that the fault recovery plan composition may result in several alternative plans to reach the same goal. In such cases, *ARP* should refer to policies in order to choose the right one which complies with the management objectives. A recovery policy can be interpreted as constraints for plan composition. Considering the previous example on EP optimization, obviously a link failure recovery plan that involves the step which uses domain *D2* to reach destination prefix *p1* could not be considered as valid, because there is a management policy which forbids using this step. In this case, *ARP* has to search for other alternatives. Another example of using high-level policies is to recover from router failure, assuming there are alternative steps involved to recover the router failure: one of them involves using router *B* as temporal fallback router, to which the traffic could be shifted to, while another step requires recalculating and applying the alternative links to route the current traffic. Obviously, both steps lead to same goal, namely, keeping the disruption on the current traffic to minimum. Nevertheless, the fallback solution maybe preferred in terms of shorter traffic disruption compared to link recalculation. If we assume that there is a policy that states "*Router B can only be used as temporal failover for A between time 20:00 - 23:00"* and the link failure on router *A* happens on *10:00*, apparently, the first alternative cannot be considered by the automated planner, because the constraint derived from the high-level policy restricts the usage of router *B* as fallback router at the time of failure on *A*.

4.3.2 KD and ARP Collaboration

KD contains a P2P network for knowledge exchange and update. A peer shares resources (e.g. failure cases and actions) with other peers and provides search facilities for other peers. A peer also bears a CBR engine for choosing best solutions. The CBR engine, upon receiving requests from *ARP*, works on the case database to propose solutions for *ARP*. The design of *KD* below explains core issues related to building P2P the network, case retrieval and reasoning in CBR regarding the focused problem domain.

Communication: *KD* uses a Gnutella-style backbone network of super peers as an appropriate overlay [11]. The overlay sticks to the characteristics of Gnutella super-peer networks. Such networks organize peers into several clusters, any cluster contains peers connected to a super peer, and the connections among super peers form an unstructured overlay network. Any peer communicates with its super peer for sending queries, publishing resources or receiving answers. Any super peer, upon joining the

overlay, has to perform several non-trivial tasks such as search and lookup, reasoning, or maintenance.

Case Retrieval: This issue involves case representation and similarity function. A case including failures and actions is represented by field-value vectors, where values are binary, numeric or symbolic. In particular, a case contains a vector v_f with symptoms sym_i for failure descriptions, a number of vectors v_a with an action *act* and conditions cnd_i for action descriptions, and a goal vector v_g with symptoms sym_i for verification descriptions. The following example explains how a case is represented:

- v_f = <sym_1:link_to_dest_failed, sym_2:no_traffic_flow, sym_3:primary_ router_not_running, sym_4: second_brouter_exist, sym_5: second_brouter_ok, sym_6: bandwidth_reserv_required>
- v_g = <sym_1: link_to_dest_ok, sym_2: traffic_flow_ok, sym_3: bandwidth_ok>
- v_a = <*act*: use_alternative_link, $cond_1$: link_to_dest_failed>
- v_a = <*act*: fallover_to_secondary_router, $cond_1$: second_brouter_exist, $cond_2$: second_brouter_ok>
- v_a = <*act*: establish_link>
- v_a = <*act*: reserve_bandwidth, $cond_1$: bandwidth_reserv_required>

Note that a request sent to the subsystem only contains v_f and v_g; the subsystem works on symptoms appearing in the request by communicating with other peers and looking into the local case database for similar symptoms and actions. Using this representation, evaluating case similarity is straightforward since it depends on comparing pairs of symptoms and values; values avoid using textual fuzzy descriptions and the expert determines the weight values of symptoms. Similar cases are evaluated by the *global similarity* method $sim(r,c) = \sum_{i=1}^{n} w_i sim(r_i, c_i)$, where *n* is the number of matched symptoms; r_i and c_i are symptoms of request *r* and case *c*, respectively; $sim(r_i, c_i)$ is the distance between r_i and c_i, w_i is a weight value of the i^{th} feature so that $\sum_{i=1}^{n} w_i = 1$ with $w_i \in [0,1], \forall i$.

Case Reasoning: This issue involves case adaptation (or reasoning, inference) and decision making. The conventional inference process carries out distinguishing a retrieved case from the problem to clarify main differences and modifying the retrieved case following the differences to obtain the final case. There are several alternative inference methods depending on problem domains, such as probabilistic inference using Bayesian networks, classification using machine learning, optimization techniques using genetic algorithms (GA). For the focused problem domain, the reasoning engine in *KD* employs optimization techniques to search for an optimal set of actions that satisfies symptoms from the request given constraints from the goal vector and conditions from the action vectors.

To enable the planning process, *ARP* needs to aggregate the recovery knowledge from *KD*. After receiving the fault report from the monitoring component, the domain analysis module dispatches the knowledge search request regarding the impacted component to the interaction processing module of *KD*. The search results are returned by *KD* in the field-value vectors format. They include knowledge such as fault symptoms and solution steps. However, the results cannot be directly used by the planner algorithm; the domain analysis module needs to convert the received data into some specific planning language, such as Plan Domain Description Language (PDDL) [11]. In the next section, we present examples on the representation of recovery domain knowledge.

As mentioned, in section 4.2 the core of this subsystem is the automated planner module. This module utilizes the Artificial Intelligence (AI)-based planning algorithm to generate plans according to the domain description. A planning problem is based on the restricted state-transition system [13]. A state-transition system Σ could be represented as $\Sigma = (S, A, E, \gamma)$, where $S = \{s_0, s_1, s_{2,\ldots}\}$ is the finite set of states; $A = \{a_0, a_1, a_{2,\ldots}\}$ is the set of actions; $E = \{e_0, e_1, e_{2,\ldots}\}$ is the set of events and $\gamma : S \times A \times E \to 2^S$ denotes a state transition function.

A planning problem can be defined as a triple $P = (\Sigma, s_0, g)$, where $s_0 \in S$ is the initial state of a problem and $g \subset S$ is the set of states which satisfy the goal. Provided with problem domain descriptions, a planning algorithm operates on the provided information and finds one or more plans accordingly. Additionally, a planning algorithm observes different constraints if they are provided. The use of constraints serves two purposes: first, it reduces the plan searching space and second, it guides the plan searching in a correct way, e.g. some of the actions should be avoided according to constraints (policies).

4.4 System Analysis – Case Study

An outbound inter-domain TE scenario is chosen to investigate recovery from EP failure using the introduced architecture. Normally, upon an EP failure, traffic is shifted to another available EP according to the BGP route selection policies. However, if a large amount of traffic is shifted, congestion is likely to occur on these new serving EPs. An intuitive approach to minimize this congestion is to redirect the traffic to another EP by adjusting BGP routing policies in an online manner until the best available EP has been found. Such online trial-and-error approaches, however, may cause router misconfiguration, unpredicted traffic disruption and BGP route flooding, leading to route instability. It is desirable to have an efficient fault recovery plan and an optimization algorithm in order to proactively produce a configuration for optimal performance under normal and failure scenarios. Therefore, we focus our interest on outbound inter-domain TE and recovery planning in case of any EP failures. Once one of the EPs fails, our recovery plan follows the proposed dual stage recovery process. First, *PBM* enforces appropriate policies (Table 4-2, P1-3) and a short-term solution is used to minimize disruption. Then *ARP*, through its interaction with *KD*, attempts to discover a long-term solution to recover from the fault. The topology for the described scenario is shown in Figure 4-2.

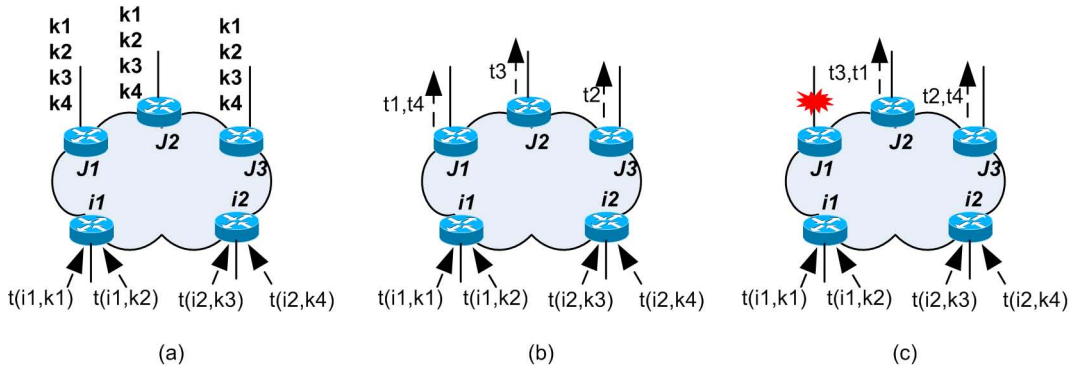


Figure 4-2. Scenario topology: (a) EP selection problem inputs, (b) PrimaryEgressPoint selection and (c) BackupEgressPoint selection if j1 fails.

4.4.1 Automated Recovery Using Outbound TE Algorithm

Based on the proposed architecture, the first step after a failure is detected is to react with a short-term solution to minimize disruption. This solution is pre-planned by executing an EP selection algorithm in advance and storing its output as policies on the Policy Repository. The algorithm's output is based on a generated synthetic inter-domain traffic matrix. The traffic matrix consists of a set of inter-domain traffic flows that originate from each ingress point towards each of the considered destination prefixes. Each inter-domain traffic flow is associated with a randomly generated bandwidth demand according to a uniform distribution. We use a sample topology and traffic flow to demonstrate the applicability of our recovery methodology. Details about the algorithm and evaluation results can be found in [9].

For a better understanding of our EP selection algorithm, we provide an example in Figure 4-2. Figure 4-2(a) illustrates the inputs for the EP selection problem, comprising ingress routers i1 and i2, EPs j1, j2 and j3, inter-domain traffic demands *t1=t(i1,k1)*, *t2=t(i1,k2)*, *t3=t(i2,k3)* and *t4=t(i2,k4*). The destination prefixes k1, k2, k3 and k4 that can be reached through all the three EPs are also shown. Recall that the task of the EP selection problem is to determine, for each destination prefix, both a PrimaryEgressPoint to be used under no failure and a BackupEgressPoint to be used when its PrimaryEgressPoint has failed. Figure 4-2(b) shows a potential solution of PrimaryEgressPoint selection, where t1 and t4 reach their destination prefix k1 and k4 respectively through EP j1, t2 reaches its destination prefix k2 through EP j3 and t3 reaches its destination prefix k3 through EP j2. This assignment corresponds to Table 4-1 column 2. In addition, Figure 4-2(c) illustrates a potential solution of BackupEgressPoint selection when EP j1 has failed. As shown, t1 has been re-assigned to EP j2 to reach k1 and t4 has been re-assigned to EP j3 to reach k4 as their BackupEgressPoint. This assignment corresponds to Table 4-1 column 3 and according to the high-level goal, this solution achieves inter-domain link load balancing.

Prefix	Primary	Backup		
		if J1 fails	if J2 fails	if J3 fails
k1	J1	J2	J1	J1
k2	J3	J3	J3	J2
k3	J2	J2	J3	J2
k4	J1	J3	J1	J1

Table 4-1.Assignment of primary and backup egress point selection.

To implement this solution, for prefix k1 the largest value of BGP local-preference, e.g. 100, should be assigned to its selected PrimaryEgressPoint (i.e. EP j1), the second largest value, e.g. 80, should be assigned to its selected BackupEgressPoint (i.e. EP j2) and any BGP local-preference value less than 80, e.g. 50, can be assigned to the remaining EPs (i.e. EP j3).Also for prefix k4 the largest value of BGP local-preference, e.g. 100, should be assigned to its selected PrimaryEgressPoint (i.e. EP j1), the second largest value, e.g. 80, should be assigned to its selected BackupEgressPoint (i.e. EP j3) and any BGP local-preference value less than 80, e.g. 50, can be assigned to the remained EP (i.e. EP j2). Moreover, since the other two prefixes reachable through j1 (i.e. k2 and k3) are assigned to j1 for neither Primary nor Backup, their BGP local preference should be any value less than 80, e.g. 50.

Egress Point	Prefix	BGP Local Preference	Policies (Event==setupEP())
J1			*if* (EP==j1)
	k1	100	*then*[(set-local-pref(k1)=Prim-val)
	k2	50	(set-local-pref(k2)=Low-val)
	k3	50	(set-local-pref(k3)=Low-val)
	k4	100	(set-local-pref(k4)=Prim-val)][P 1]
J2			*if* (EP==j2)
	k1	80	*then*[(set-local-pref(k1)=Back-val)
	k2	80	(set-local-pref(k2)=Back-val)
	k3	100	(set-local-pref(k3)=Prim-val)
	k4	50	(set-local-pref(k4)=Low-val)][P 2]
J3			*if* (EP==j3)
	k1	50	*then*[(set-local-pref(k1)=Low-val)
	k2	100	(set-local-pref(k2)=Prim-val)
	k3	80	(set-local-pref(k3)=Back-val)
	k4	80	(set-local-pref(k4)=Back-val)][P 2]
Initial configuration policy			
Event==BGP-conf if (-) then [(set-local-prefs([Prim-val,Back-val,Low-val],[100,80,50])), gen-event-setupEP(j1,j2,j3)] [P0]			

Table 4-2. Local-pref setting and policies for prefixes on egress points.

Table 4-2 shows the assignment of BGP local-preference setting for prefixes k1 to k4 on all EPs. These configuration settings are enforced on the EPs by their PEP based on policies (P1, P2, P3). An initial configuration policy (P0) is used by the network administrator, to configure the proper values for BGP local-preference.

The benefit of combining a TE algorithm with a PBM approach is the creation of a flexible management environment able to quickly react to failures. In parallel, our framework automatically works to recover from the failure and recovery policies are used as input to the *ARP* subsystem. With the cooperation of *KD*, *ARP* outputs a new recovery plan that *PBM* can enforce to the network and reinstate normal operation.

4.4.2 Automated Recovery Using Planning Algorithm

To show how the planning and knowledge discovery subsystems collaborate, we present here an example based on the aforementioned border router (EP) failure

scenario. The objective of the collaboration between the two subsystems is to produce long-term fault recovery solutions.

After the router failure is recognized, the first stage recovery procedure is activated in order to sustain the current traffic and minimize the disruption of the failure. Whereas such a solution can be regarded as a short-term measure, a long-term recovery measure is still needed to completely recover from the failure and reinstate normal operation.

The long-term recovery process is activated by the reports from the monitoring component. The domain analysis module analyzes the monitoring report and extracts the current global state. The current global state includes the information on the impacted component and other relevant information. Note that the granularity of monitoring information will affect the later planning process. For example, a report that states *The primary router is impacted by failure and it has OS version 2.3* will be more useful for planner to compose a better plan than one that just states *The primary router is impacted*. The global states are represented by the set of predicates.

Algo. 1: Automated Planning
Input: O: set of actions
s_0, g: initial state, goal state
Output: π sequence of actions
1 $s \leftarrow s_0$
2 $\pi \leftarrow \emptyset$
3 while True do
4 if s satisfies g then
5 return π
6 $A \leftarrow \{a \mid a \in O$ & precond(a) true in $s\}$
7 applicable $\leftarrow A$
8 if applicable$= \emptyset$ then
9 return failure
10 choose $a \in$ applicable
11 $s \leftarrow \gamma(s, a)$
12 $\pi \leftarrow \pi.a$

Algo. 2: Action Optimization
Input: C: set of retrieved cases c_i;
R, τ: request with symptoms r_j, threshold
Output: σ optimal set of actions
1 $\Sigma \leftarrow \emptyset$
2 $\Sigma^* \leftarrow \emptyset$
3 for each $c_i \in C$ & $r_j \in R$ do
4 $\sigma \leftarrow \{a \mid a \in c_i$ & a satisfies any $r_j \in R\}$
5 $\Sigma \leftarrow \Sigma \cup \sigma$
6 while True do
7 for each $\sigma \in \Sigma$ do
8 evaluate $f(\sigma)$
9 if σ is optimal then
10 return σ
11 $\Sigma^* \leftarrow \{\sigma \mid \sigma \in \Sigma$ & $f(\sigma) > \tau$ }
12 $\Sigma \leftarrow \{\sigma \mid \sigma$ generalized by σ^* & $\sigma^* \in \Sigma^*\}$

After the failure source is known, the domain analysis module dispatches a search request to *KD* in order to find possible resolution steps to recover the component. *KD* collects solutions from various sources and then runs the reasoning engine on the retrieved solutions before returning the proposed actions to the analysis module. The reasoning engine uses the optimization technique based on the GA algorithm to provide an optimal set of actions, see Algo. 2. The discovered recovery actions have to be translated into a planning-specific language. For example, if a recovery action involves upgrading the router OS, this action could be described in PDDL [11] as:

```
(:action fetch_update
  :parameters(?r - router ?p - patch ?cv - currentversion
              ?nv - latest version )
  :precondition( and ( (failed ?r) (patch_at ?r ?p ?cv)
                                   ( < (?cv ?nv)  ) ) )
  :effect( and ( (updated ?r ?nv ?p) (= (?cv ?nv) ) ) ) )
```

The *parameter* field denotes which parameters are needed for this action. The field *precondition* describes the conditions, under which the *fetch_update* action could be applied. The *effect* field denotes the consequence of this action, i.e. how this action is going to affect the global state.

The question mark denotes the variables. The initial state, goal state and recovery options formulate a planning domain. The automated planner operates on the planning domain for one or more feasible plans. The planning algorithm [13] is described by Algo. 1. Note that only those actions ($a \in O$) which cause state changes and the current state *s* are considered as parameters of γ function in this algorithm, events are left out for the sake of simplicity.

Recovery actions are selected based on their preconditions and current global state. Each action leads the current global state into a new state, this iteration finishing when the new state equals goal state. The generated plan is a sequence of actions which transit the initial state into the goal state.

4.5 Related Work

The study of Mark et al. [14] has proposed an automatic system for finding known software problems. The system matches the symptoms of the current problems with the symptom database to find the closest matches. The matching algorithm is designed to work with structured symptoms that contain programs called stacks, not arbitrary data. The recent study of Stefania et al. [15] has proposed a CBR system for self-healing in software systems. The description of the system lacks some details. Any case is represented in features which contain binary and symbolic values. Cases are retrieved by using the k-NN (nearest neighbours) algorithm; where the similarity distance function evaluates case features with the corresponding weight values, but the weight function for features is not provided. The system mainly depends on the retrieval process to classify problems.

Policy Based Management (PBM) simplifies the complex management tasks of large scale systems, since high-level policies monitor the network and automatically enforce appropriate actions [1], [16], [17]. Industry and PBM have been closely related in autonomic computing and self-management approaches [18]. The main advantage, which makes a policy based system attractive, is the functionality to add controlled programmability in the management system without compromising its overall security and integrity. Policies can be viewed as the means to extend the functionality of a system dynamically and in real time in combination with its pre-existing hard-wired management logic [1], [17]. Policies are introduced to the system and parameterized in real time, based on management goals and gathered information. Policy decisions generate appropriate actions on the fly to realize and enforce those goals.

Outbound inter-Autonomous System (AS) Traffic Engineering [9], [10], [6] is a set of techniques for controlling inter-AS traffic exiting an AS by assigning the traffic to the best egress points (i.e. routers or links from which the traffic is forwarded to adjacent ASes towards destinations). The general problem formulation of outbound TE is: given the network topology, BGP routing information and inter-domain Traffic Matrix (TM), determine the best Egress Point (EP) for each traffic demand so as to optimize the overall network performance, such as inter-AS link load balancing [6].

Kephart [18] addresses in his study the challenges of using AI-based planning methods to the autonomic computing. Srivastava et al. [19] discussed the feasibility and theoretical aspects on using planning methods in autonomic computing. They concluded that automated planning is an evolutionary next step for autonomic systems that possess the self-managing capabilities. Arshad et al. [20] presented a planning based recovery system for distributed systems. However the proposed approach has several disadvantages; e.g. the recovery knowledge elicitation is not considered and

lack of details in many aspects of applying automated planning methods in fault recovery.

4.6 Conclusions and Future work

Motivated by the need for an efficient fault recovery management, we have presented our initial efforts towards an integrated architecture. By combining the strengths of three paradigms; Automated Recovery Planning, Knowledge Discovery and Policy based Management, we attempt to provide an automated framework. We have demonstrated a dual stage recovery process based on a case study of border router (EP) failure, aiming to maintain optimized configuration of outbound inter-domain traffic and quickly reinstate normal operation.

Beyond initial architecture design, we intent to investigate further interactions among subsystems and define generic and reusable interfaces. This will allow the extension of the architecture for a variety of case studies. We will aim to increase the automation of recovery and plan execution, thus minimizing human intervention and recovery times. Our future work will focus on intelligent planning algorithms that will combine system knowledge and business goals, aiming to gradually migrate to fully automated fault recovery management

5 Context Architectures for Autonomic Management (CARAM)

5.1 Introduction

Trends in communication and information technologies propose interoperable environments where the information is being managed between applications from different technologies and shared by operators of the information in a transparent manner [21], [22]. This complex and interoperable panorama of the information acts as the trigger in the development of new functional service-driven architectures over next generation networks based on changes from business goals (service context) and variations in the resource environment (resource context). The perspectives in the design of new systems managing context information set as a challenge and as a main requirement the premise where context information must be perfectly captured and gathered, even in real time, for executing service operations. Furthermore, communications demands for context information require the information to be used as an interoperable information source [23], [24].

Context information comes from various sources even more so in ubiquitous environments and pervasive applications where the information is dynamic and changes rapidly. Thus, context information must be gathered and used in different provider and user domains involving activities such as modelling, exchange protocols, trust in the information, and other information properties such as consistency and resilience and many more. So far different context modelling approaches have been proposed [23], [24], [25], [26], [27], [28], [29] and typically context management is separated from the implementation task of applications by using frameworks and the like.

On the one hand, special emphasis is put on research challenges from the next generation networks for supporting services with the introduction of context information. On the other hand, requirements for managing and composing context information in

the process of context provisioning are derived. The main objective of this work is a new concept for context models suitable for representing information and managing services in cross-layered environments; it integrates the work from the participating partners in the CArAM research project. The policy based management paradigm is used for supporting cross-layered service interoperability in autonomic computing environments and the integration of context information in management operations of pervasive services. The model makes use of the DEN-ng information model and ontologies. The final aim is to extend the functionality of the model to previous solutions for the provisioning of context information as devoted in the context composition (CoCo) infrastructure. In this work we describe the research challenges of the CArAM project and initial results.

The combination of policies and context information in a formal way for supporting and managing services has been studied in previous works [30], but such studies leave aside the context information provisioning, as it was assumed that certain information for triggering the services management operations were already captured and composed with the necessary format for facilitating interoperability in exchanging information between cross-layered applications. In this work we aim to go one step further and present a functional architecture that is intended to support context provisioning for providing context information from context information services (CIS) e.g., content providers, service providers and network operators databases, and thus facilitate the control of pervasive service life cycle.

Figure 5-1 depicts the CArAM project scope; key elements involved in the context provisioning that are also described in this work are the Context Performer, Ontology-Based Context Interactor and the Context Reasoner. These three components provide the framework for supporting information interactions with Context Information Systems as a result of the integrated CoCo Infrastructure as one more crucial component. The core idea is to provide an intermediary system to act as a middleware solution between policy based service management systems and diverse context information systems, providing an ontology-based solution. An architecture interacting with ontology-based models supports information interoperability in order for the exchange of information and the policy based management activity can be performed transparently.

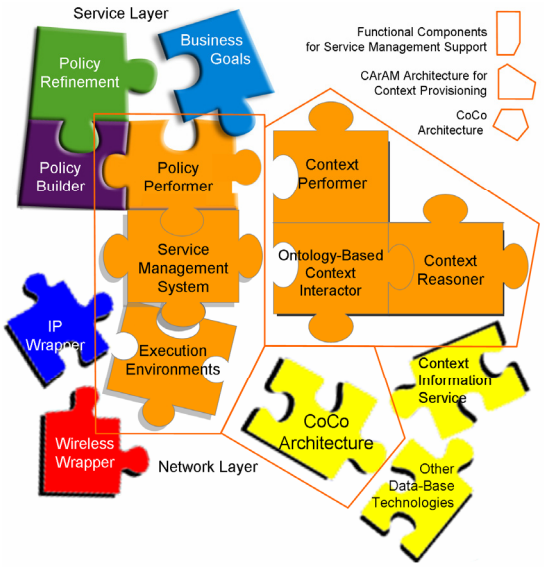


Figure 5-1. Functional components for context provisioning and services support.

The remainder of this section is organized as follows: section 5.2 reviews the relevant research project objectives and challenges for supporting services with the introduction of context information, as well as context information requirements for managing and composition in the context provisioning task. Section 5.3 describes the general CArAM functional architecture, research approach and the future plans for implementation. Section 5.4 illustrates the CArAM expected contributions beyond the state-of-the-art. Finally, we present concluding remarks and outline future work.

5.2 Objectives of the CArAM Research Project

Following the idea of next generation networks highlighting the importance of service awareness, this approach manages context information provisioning for facilitating the management of the full life cycle of a service by using ontology-based policies, which represent the desired semantics to be enforced.

In the CArAM research project the context information is used in policies as information necessary for triggering actions and control service dependencies. This enables management frameworks to take into account the variations in context information (contained in service management policies), and relate those variations to changes in the services operation and performance of the networks (e.g. Quality of Service (QoS) and network performance). The synergy obtained from the context composition, the ontology-based data representations and the policy based paradigm represent the knowledge engineering that autonomic systems provide [21].

5.2.1 Research and Scientific Challenges

From the perspective of this research, context provisioning is the complete collection of information, its explicit composition with certain syntax and the parsing action to build information classes as a result of a formal process. The formalized context information is then able to trigger actions as it is considered as events or values in policies across management boundaries and abstraction levels in order to achieve specific goal(s) or behaviour(s). Once context information is formalized, policies from all interacting management domains can affect each other as information is used and is available to be shared as knowledge capsules.

5.2.1.1 Context Provisioning

In our approach, the policy based management of autonomic communications relies on the correct capturing of relevant information about the network called its context. Generally speaking, context information can be any information that characterizes the situation of an entity [31]. Context Provisioning comprises all phases dealing with context information from the acquisition of the latter at sensors to its offering to applications. Context information can thereby come from various sources with different interfaces like e.g. physical sensors, databases or GPS receivers, and may not be interpretable by a context-aware-service (CAS) right away. In addition, those devices have very different computing power and storage facilities, and thus it is useful to abstract from real sensors and refine and possibly store context information in a context provisioning middleware. Over the last years, there have been many different approaches to the management of context [32], [33], [34], [35]. In the CArAM project, we will use the CoCo middleware that is based on Semantic Web technology [36].

After context sensing, the “raw” information can pass through different refinement steps like transformation, combination, aggregation and the extension with meta information.

Finally, the relevant context information is provided to the requesting CAS to be used in the contextualization process.

5.2.1.1.1 Context Modelling

As context information is gathered from many sources and has to be useable in various CASs an information model is crucial. Research in this area has included key-value-pair approaches, the use of markup or graphical languages, logic-based models, object-oriented descriptions and the currently very popular use of ontologies [37]. Ontologies combine a formal foundation with good extensibility und usability and allow for the derivation of new knowledge by inference.

The CoCo middleware uses an ontology-based information model called Context Meta-Model (CMM) [38] to describe context both internally and at the provisioning interface. This is a generic context modelling approach that defines a common meta-model and allows for domain-specific context models. For the CArAM project we will thus develop a suitable context model to use by CoCo. This context model should interact in the best possible way with the Ontologies for Support and Management (OSM) / DEN-ng approach applied in the Onto-CONTEXT part, see Figure 5-3.

5.2.1.1.2 Context Refinement and Composition

As mentioned above, the data gathered at sensors is not always directly usable. A simple example for a format conversion is the transformation of location information from World Geodetic System (WGS-84) coordinates to Gauß-Krüger coordinates. More complex operations in the refinement process include the combination and aggregation of different context information. For instance, the video conferencing service in the meeting room has to start automatically. This can be derived by the combination of two information sources. The calendar application knows when a meeting is scheduled and the person tracking system in the office building can tell when all people are in the meeting room. Also, meta-information like quality parameters and other are important for context management, too. E.g. historical context information is helpful to many services; therefore the time of context sensing is crucial to many uses. For the CArAM project, we have to define a suitable subset of context parameters and possible sources. We will thus find refinement and composition processes necessary.

5.2.1.1.3 Context Managing

The same context information can be used for multiple services and depending on the service, context information that is not valid any more is still important and has to be stored in the context architecture. For the CArAM project we will build on the CoCo Infrastructure that already applies technologies to store Resource Description Framework (RDF) and Web Ontology Language (OWL) information as well as provides an inference engine. But so far, the deployed technology for this task has not shown good performance and should thus be replaced. Also, for the storage of context information a suitable algorithm should be implemented to delete context information from the cache that is not needed any more. The CArAM project will also focus on the inference of new knowledge from available context information to enhance the quality of information used in CArAM.

5.2.1.2 Autonomic Management

Business-level objectives are represented by high-level policies that define desired behaviour for all system elements. Often, four different aspects of self-management:

self-configuration, self-optimization, self-healing, and self-protection are cited. The CArAM approach is focused on providing a solution that enables multiple policy based systems using their own, different, policy information models to interoperate with each other. Thus a middleware solution must be deployed in service level planes. This solution is offered by using ontologies for semantic integration, and it is triggered by events signalled between the involved domains. Management systems in different service levels follow autonomic principles.

5.2.1.2.1 Self-Management

Self-management functions imply the realization of simple operations in a more dynamic form and increasing the level of automation of the intelligent control loop described as 'Collect–Analyze/Decide-Enforce/Change'. 'Collect' is about getting a value from a specific database that allows triggering a service about the surrounding environment in order to build a self-awareness application; 'Analyze/Decide' involves inference if the information can be related with other information from other systems. This process of diagnosing the problem is based on the collected information and the references in the model whether the information can be transformed to another certain format. 'Enforce/Change' comprises deployment, which adds references to build new components, and define configurations, which change the existing functionality by means of programmability of systems. This intelligent loop eventually leads to a distributed, adaptive, global evolvable system capable of fostering continuous changes as it depicted in Figure 5-2.

Self-management has recently acquired importance from technology focuses on network element granularity. Multiple systems interact automatically and dynamically for building distributed systems with certain clarity in their management processes. CArAM aims to provide solutions to the next key challenges:

- Developing a middleware framework for CIS's containing enough semantic knowledge in the environment model in order to correlate all necessary information and thus optimize service management components. The framework thereby uses a wide variety of global context information services.
- Developing the orchestration, interworking and integration between separate self-management functions both at database element and global management levels.

5.2.1.2.2 Policy Based Management

As far as the PBM is concerned, CArAM work is not just to apply existing PBM techniques to the network aspect of pervasive services or to concentrate on the building-up or extending the standard-based information model as most do (e.g., Policy Core Information Model by IETF, Core Information Model by DMTF, Parlay policy management APIs).

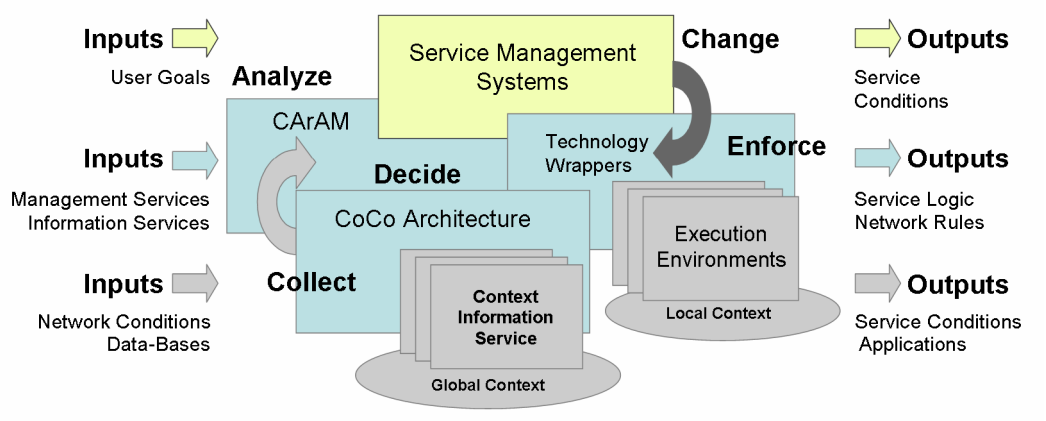


Figure 5-2. CArAM self-management approach.

While being aware of the importance of these two aspects of PBM research, CArAM puts specific emphasis on the policy based description of pervasive services, and a practically functioning pervasive service information model using ontologies to do so. The fact that there is not any reference implementation for the IETF Policy Core Information Model (PCIM) or the DMTF Common Information Model (CIM) makes the PBM hard to put into practice. So we go one step further and we define as a CArAM objective the enhancement and control of the full service life cycle by means of policies; in addition, CArAM takes into account the variation in context information, and relates those variations to changes in the services operation and performance inspired from autonomic management principles.

5.2.1.3 Orchestration

Next generation network and services requires the optimization of multiple types of orchestration mechanisms, enabling network operations (i.e. optimization for initialization, dynamic reconfiguration, adaptation and contextualization, dynamic service deployment support) and service tasks to be optimized. In the CArAM research project, new primitives and a self-management mechanism acting as intelligent intermediaries between service management entities and the network and context information services is being considered.

5.2.2 Technological Challenges

This section describes the most important technological challenges that we foresee in the CArAM project.

5.2.2.1 Define an Autonomic Architectural Solution for Autonomic Management

Architectural design process needs to take into account requirements of specifically selected context representation – particularly at abstract information level, though the implementation requires a well thought out data representation. The way context information is handled and stored by CoCo Infrastructure needs to be translated into management systems compatible parameters in order to create proper triggers for appropriate management actions. This sets up the basic requirements for implementation tasks that are to be carried out as the part of CArAM project. As CoCo Infrastructure operates on RDF / OWL, a proper information parser needs to be implemented in order to construct a meta structure of context information that can be easily transformed into objects understandable by the management plane. Such meta-

structure needs to be carefully selected so it can be regarded easily scalable, as flexible as possible and easily extensible.

5.2.2.2 Reuse and Integration of Information Resources

Since context information can come from different sources with different properties, all natural consequences of such information source diversity need to be taken into consideration when planning development work. Information needs to be pre-processed before storing it in a context information database in order to qualify it as appropriately reliable, up-to-date and complete. In order to minimize unnecessary resource consumption, information ought to be cached and any additional derivative information should be created reusing as much data already in the database as possible, together with any required new data, as stated in section 5.2.1.1.2. As the middleware must be able to work on data coming from different management domains, the necessity of integrating various management related databases arises. The software should be capable of handling different types of management data held in storage to fulfil basic interoperability requirements.

5.2.2.3 An Autonomic Knowledge Plane to Support Service-oriented Systems

The CArAM project will make use of a single information model for management and support (OSM) that will serve to capture the necessary entities concerned with service-oriented resource orchestration and their relationships [39], [40]. From OSM information model, a set of data models will be derived that will enable the concepts to be mapped to specific standards and technologies. In addition, a set of ontologies will be created that extend the semantic base of the models to enable behaviour orchestration and new entity relationships. The knowledge plane supported on OSM ontology allows the representation of heterogeneous devices, the construction of appropriate behavioural instructions, and the realization of a collective cognition across resources. This will go some way towards facilitating the representation of context information used to drive the services life cycle in an autonomic manner.

5.2.2.4 Programmatic Enablers for Autonomic Service-driven Configuration

The CArAM project proposes the definition of a framework for dynamic (de-) activation and reconfiguration of new and existing context information services. Context from management information services with different database technologies as well as from network components will be integrated transparently. It will help service management designers to go from “dumb black boxes” to autonomic network resources and gather context information. CArAM can be seen as a set of programmable enablers for dynamic services regarding context information for new configuration activation and execution into network elements to support new service functionalities at runtime.

5.3 CArAM: Framework and Architecture

The CArAM research project takes input from diverse initiatives regarding context composition, service management and ontology engineering to create an appropriate autonomic control loop supported by an architectural framework that focuses specifically on the issue of a self-managing service driven context-resource orchestration. The CArAM autonomic framework will seek to automate the context-resource orchestration process via a policy-driven closed loop control process that makes decisions using knowledge supported by decisions made from context information and data models, augmented using ontological data.

The objective of this research is to create an automated and domain independent approach to select policy parameters. Policy parameters are considered as context information, for service management policies (e.g., service deployment) such that the corresponding policy goals are met. The approach is supported by a novel combination of ontology engineering and context composition solution. The added value from this approach is the reduced reliance on technological dependencies for services support and increase of the interoperability between heterogeneous policy based service management systems providing the technological support for the context provisioning.

The framework depicted in Figure 5-3 provides the overall objective for our research project. CArAM is based on the use of formal ontology languages like OWL [41] for endowing with the formal mechanisms to service management systems and thus context information, necessary in task as service support, are managed and used for service management operations in the service execution. This interaction relies on supporting ontologies using some ontology-based components.

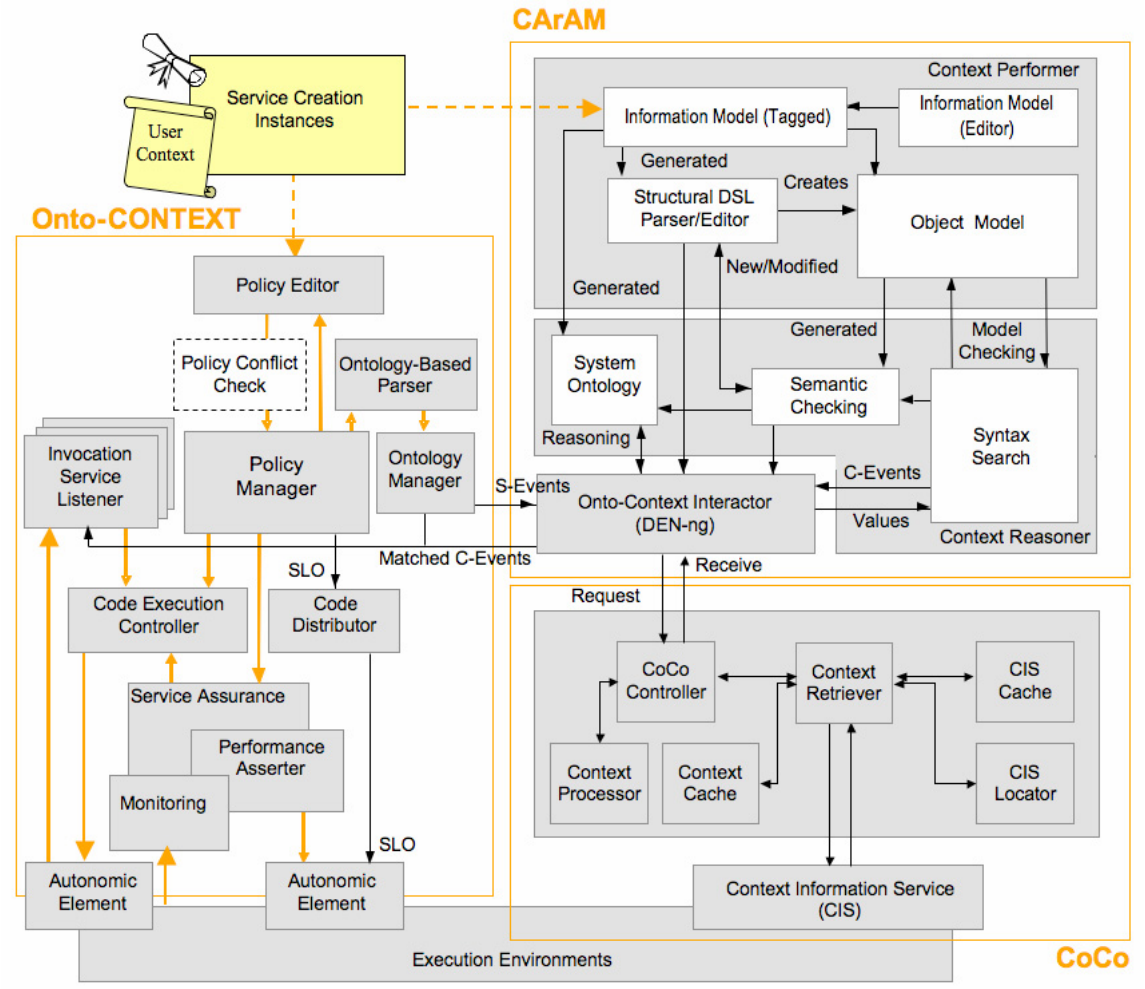


Figure 5-3. CArAM architecture.

5.3.1 CArAM Architecture

In the CArAM architecture the “Ontological Comparison” refers to concepts and relationships matching between objects defined in the OSM policy information model and the CMM used by CoCo, enabling the *Ontology-Based Context Interactor* for reasoning about received data and subsequently infer meaning(s). For example, the *Onto-Context Interactor* requests information at the CoCo interface. Inside the CoCo

Infrastructure, the *CoCo Retriever* will check whether this information is available in the CI Cache. If it cannot be retrieved from there, the *CoCo Retriever* will query a *CIS* and return the information via the *CoCo Controller* to the *Onto-Context Interactor* (for more details see section 5.4.2). Once the information has been received we can infer which users, customers or components from a certain application are affected as result of a *Syntax Search* for looking keywords related with policies. *Ontology Managers* always will use the information contained in system ontology pertaining by looking at relationships in both the information model and relevant ontologies and deducing which customers are affected by the information received.

The *Context Reasoner* determines whether the actual state of the managed entity is equal to its desired state (contained in a policy). This is performed as a result of the model checking and the *Semantic Checking* process. Thus the *System Ontology* helps to confirm the integrity of links and relationships on monitoring the received value entity. If it is not corresponding, then the *Semantic Checking* prepares the set of configuration changes that should be sent to the *Parser/Editor* to be modified or to the *Onto-Context Interactor* to be once requested.

This “translation” between values and C-Events is the result of logic-based rules contained by the *Onto-context Interactor*. The *Onto-context Interactor* provides the novel ability to change the different control functions (such as what context values data will be used to trigger services in a form of C-Events) to best suit the needs of the changing environment and relating C-Events with S-Events. Thus Matched C-Events are used to send the Invocation Service to Listeners from Service Management Systems ready to process such conditions in a now re-formatted instruction containing the event value entity.

5.3.2 Management of Multiple CISs

Besides the interface for ingoing context requests, the CoCo Infrastructure has a second interface to query CISs for context information. In general, this is a classical service discovery problem but here the difference is that a service is necessary so as to deliver context information for a specific entity at a given moment in time. To minimize both delay and costs, a multi-level approach is taken. The CoCo Infrastructure caches acquired context information for later use. The raw data is stored in an RDF-based context cache and can possibly be used multiple times as current description of the environment, to derive new knowledge, and later also as historical data. Second, a so-called CIS Cache stores the semantic service description of known CISs and the CIS Locator can search by means of a declarative description for new CISs. To further enhance the quality and accelerate the process, statistics about past queries and their success might be helpful additionally.

5.3.3 Context Information Services

Based on the services to support, varying sensors have to be deployed to gather the needed context. In previous work, the CoCo Infrastructure has queried different kinds of context sources, e.g. various Web Services, physical sensors for temperature, brightness and humidity measurements, and local databases. Both one-time requests for information as well as subscriptions have to be supported.

5.3.4 Extending Information Models and Deriving New Knowledge

As part of the CArAM research project, a policy-context information model will be used. The policy-context information model is represented with ontologies to capture the

concepts relating to the context-driven self-management of services. However a set of data models will be derived from this information model that pertains to the specific technologies and standards that will be used as part of CArAM project. In addition, an appropriate set of ontologies will be used to extend model knowledge with semantics to enable reasoning amongst heterogeneous resource execution environments. Currently, the standard reasoner included in the Jena Semantic Web Framework (http://jena.sourceforge.net) is used to derive new knowledge in the CoCo infrastructure. Due to some issues this solution is rather slow and inconvenient, thus the integration of another reasoner may be preferable.

5.4 CArAM Beyond State-of-the-Art

CArAM research project aims to provide solutions to the key challenge in self-management. Self-management means that network information distributed in databases follows autonomous flows of control triggered and moderated by network events or changes in network context. The network information is self-organized in the sense that it automatically is related to available context necessary for services support, and provides the required context-matched information and furthermore any other necessary information to support requested services for self-adaptation when services are operating in response to context changes.

Figure 5-4 illustrates the mapping from research challenges to technological objectives, on to the relevant state-of-the-art and the innovation that CArAM project is going to achieve beyond.

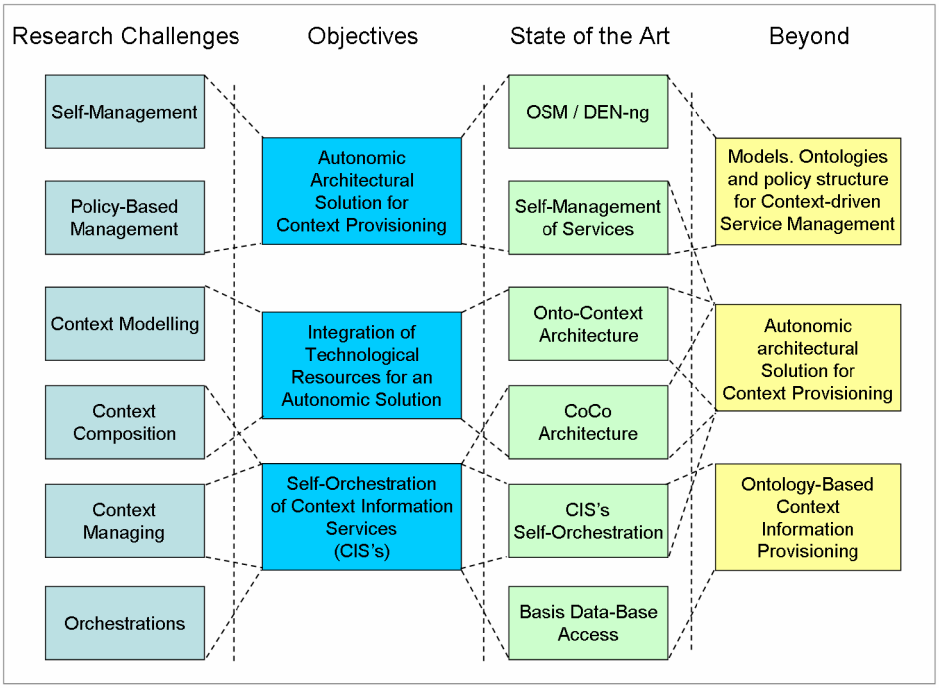


Figure 5-4. Challenges, objectives, state of the art and beyond.

5.4.1 Autonomic Computing (IBM) and Autonomic Communications (Motorola)

Modern computer science – especially computer networking – is suffering from rapid growth of complexity in architecture, interaction and relations between systems while their functionality increases [42]. As services offered by new systems are becoming more sophisticated, the underlying infrastructure requires more maintenance effort to

keep itself operating continuously preserving high reliability and performance indicators. This is especially true in case of pervasive / ubiquitous networks. Complexity growth means therefore dramatic increase in design, implementation and operation costs. This problem forced researchers to look for an alternative way of managing complicated structures of modern networks, which resulted in the idea of shifting management responsibility from network operator to the network itself by implementing self-configuration, self-optimization, self-protection and self-healing mechanisms in relevant hardware.

The CArAM project follows this trend by enhancing the efficiency of network management through an improved use of context for the purpose of management policies dynamic adjustment. As context information itself is essential for systems to be able to act according to autonomic management paradigm, it is crucial that context information provided is of best possible accuracy and as up-to-date as possible. Providing most accurate and complete data requires gathering information from various domains, appropriate data processing and presentation to the management layer. Therefore, proposed solutions must be highly interoperable and flexible.

5.4.2 CoCo Infrastructure

The CoCo Infrastructure (see Figure 5-3) acts as a broker between context-aware services that request context information and context information services (CIS) that provide context information. It therefore relieves the CAS from the burden of discovering context information services, data transformation, or derivation of high-level context information from low-level context information. CoCo stands for “composing context” and the CoCo infrastructure supports the request for composed context information. So-called CoCo graphs can describe such compositions; the graph-like structures are expressed in XML.

Basically a CoCo Graph is made out of two types of nodes: Factory Nodes describing the requested piece of context information and Operator Nodes containing instructions how to process one or several pieces of context information. The procedure is as follows: First, the request in form of a CoCo graph is sent from the CAS to the CoCo infrastructure where it is analyzed by the CoCo Controller. For each Operator Node, the controller instructs the Context Processor to execute its operation, e.g. adaptation, selection or aggregation. For every Factory Node it sends a request for context information to the Context Retriever, which at first queries the internal Context Cache, whether it already has got the requested information. In case it is not available there, the CIS Cache is asked whether it already knows an appropriate context information service to retrieve the information from. If not it then instructs the CIS Locator to find appropriate context information services, to which the retriever sends related context requests. After having received context information for each request the retriever matches this information against the request of the controller, selects the most appropriate piece of context information and returns it to the controller.

5.4.3 Onto-CONTEXT Architecture

This section provides a brief review of the Onto-CONTEXT framework [43] with the objective to validate the CArAM architecture in terms of a suitable solution for context provisioning in policy based context-aware architectures [30]. Onto-CONTEXT deals with the creation and modelling of services in order to provide service assurance and management functions by constructing a framework using autonomic elements [44] for management and execution of services.

The framework allows the use of appropriate APIs for deploying the services in mobile and IP domains. The context-policy model used is flexible and it accommodates the value of context information that needs to be evaluated, especially if the context changes. The context-policy model is based on OSM-DEN-ng, moreover, the policy model scales with the network or applications that use it. This context-policy model is expected to be accessible from autonomic elements that can reside in heterogeneous networks and architectures. Storage and retrieval of this information is also important; this requirement is met by being defined and implemented using an object-oriented philosophy.

5.4.4 Information Modelling (OSM / DEN-ng)

The OSM information model is used to model the important concepts pertaining to an environment under management perspective. The main objective of the OSM information model is to formalize a description of the problem domain outside of constraints of actual implementations.

OSM follows DEN-ng principles and defines a UML-compliant object oriented information model for the purposes of managed entities and their corresponding relationships. DEN-ng principles guide to derive multiple data models enabling heterogeneity across the management information of diverse technologies [45]. OSM augments the information and data models by using ontologies. The models alone cannot fully provide the necessary types of semantics that are needed to represent behavior and its orchestrations. The ontologies define interoperable groups of definitions, meaning and complex relationships for enabling the creation of new and different facts, facts that cannot be derived from just the monitored context data itself.

5.5 Conclusions and Future Work

We proposed to use context information to stimulate policy based management for autonomic communications. Both context information and policies are based on ontologies and thus semantical descriptions can be harmonized to foster better interoperability. All parts necessary for this transition to the self-management in next generation networks have been described in great detail and a functional architecture for the integration has been presented.

As the CArAM project is not finished yet, we still have to do part of the implementation work. From this we will learn whether our approach is efficiently fulfilling the required tasks and possibly learn whether it is necessary to make smaller refinements to our architecture. Another important, but future area of research is to add learning and reasoning algorithms to the Onto-Context Interactor with the objective for enabling the system with ability to generate “hypotheses” (infer context locations) to a more efficiently determination of context origin.

6 Distributed and Autonomic Real-time Traffic Analysis (DATTA)

6.1 Introduction

The constantly increasing data rates available in the core networks of ISPs and the increasing number of concurrent flows running through the network became a challenge for traditional centralized IP traffic analysis systems. The main purpose of DATTA

activity was to design distributed and autonomic mechanisms suitable for real-time IP traffic analysis on (very) high-speed network links. The following three different aspects of distributed traffic analysis have been considered in within the DATTA activity:

- **Distributed Real-Time Traffic Analysis**

  Real-time IP traffic analysis on high speed links is very challenging for traditional centralized solutions, mainly due to the little time available to process a packet (in the order of nanoseconds for 10 Gbps links). Many traffic analysis tasks (such as flow accounting, Intrusion Detection Systems (IDS)) are stateful operations which require state lookups for every analyzed packet thus increasing the time required to analyze a packet. Distributing the traffic analysis process to multiple nodes improves the scalability of analysis due to at least two reasons: more processing nodes are available (which allows parallel analysis of packets) and flow states may be distributed to the different processing nodes (thus reducing the state lookup time for every packet). Thus, the first task of this project is to develop fully distributed and autonomous P2P-based management mechanisms to allow self-configuration and self-healing of the analysis platform in case processing nodes join or leave the network.

- **Distributed Flow Collection**

  Centralized approaches to NetFlow [46] records collection have a big disadvantage of taking a long time to retrieve traffic with certain characteristics because they need to inspect huge NetFlow records databases. A distributed flow collection solution consisting of several collectors, each responsible with storage of a subset of flow records allows such flow queries to be answered faster while also allowing more flow records to be stored (thus having the flow records accessible for longer periods of time). Thus, the second task of this project is to design distributed and autonomous P2P-based mechanisms supporting scalability and data consistency of flow collection and to allow self-configuration and self-healing in case storage nodes join or leave the network.

- **Automated Trace Anonymization**

  Although the majority of National Research and Education Networks (NRENs) in Europe capture traffic traces to be used in research activities, having access to such traces often creates an administrative overhead caused by data protection laws. An automatic traces collection platform aims to ease the process of accessing traffic traces collected at different locations by automatically anonymizing traffic traces and by standardizing the processes of requesting and accessing of those traces. Thus, the third task of this project is to design an interface that allows to export NetFlow records (or a subset that matches some criteria) in anonymized form to any research party in the consortium. The requesting interface could be a web-service and the NetFlow records could be sent in NetFlow format to a collector specified in the request.

The three different aspects described above led to the starting of three different research projects:

- **Distributed Packet Capturing – DiCAP**

  One of the key operations in analysis of IP traffic is capturing the traffic from a network link. As link bandwidths continuously increased over the last decade at a higher rate compared to CPU and memory speed, having a single PC capturing

and analyzing traffic became unfeasible for high-speed links. Sampling is seen as one solution to leverage this gap as proposed by [48], [49], [50], [51]. By sampling, a traffic analysis system (such as Intrusion Detection Systems or NetFlow exporting device) only looks at a fraction of packets and makes its calculations and decisions based on that set. The main drawback of sampling is the decrease in accuracy of the results for the system using sampling. Other papers such as [52] and [47] have shown that packet sampling decreases the accuracy of analysis results. In those particular cases it was shown that sampling degrades the quality of the IDS algorithms investigated.

In order to address the problem of sampling in high-speed networks, DiCAP distributes the analysis work to multiple devices in order to decrease the amount of sampling used. During early investigations of this work it was observed that by using a standard PC and Linux packet capture libraries the capturing performance for links with high packet rates was very limited. Therefore, DiCAP was designed in order to cope with the high packet rates by distributing the traffic load between several PCs which capture packets in parallel.

The DiCAP architecture, implementation details as well as its evaluation are presented in [53].

- **Distributed Storage for IP Flow Records – DIPStorage**

Depending on the granularity of the data collected by network monitoring tools as well as on the type of link on which traffic is measured, the information gathered may range from a few kilobytes per hour up to gigabytes per second.

Today, traffic processing is mostly done by centralized components. Since the traffic volumes increase, centralized components no longer can cope completely with this traffic. Therefore, another strategy for processing such volumes in the future is needed. Flow accounting is concerned with measuring the IP traffic on a per flow basis. A flow is typically identified as a unidirectional sequence of packets from one source to one destination point having one or several common IP header fields. The most commonly used characteristics to identify a flow are: source IP address, destination IP address, IP protocol number, source port, and destination port. Storage of IP flow records is very challenging, mainly because a retrieval operation typically is preceded by a complex search operation, thus distributing the storage (and the search operations as well) should greatly improve the performance of a storage platform. The main goal of this work was to investigate whether P2P mechanisms can be used to improve the storage scalability and query performance of IP flow storage. In order to achieve this goal a prototype was designed, implemented, and evaluated. Especially, the approach was termed DIPStorage (Distributed IP flow Storage) to test and validate the design done. As the evaluation shows, DIPStorage meets the desired goals.

The DIPStorage architecture, implementation details as well as its evaluation are presented in [54].

- **Shared Storage Repository for NetFlow Records – NFShare**

The key aim of this project is the design and prototypical implementation of a distributed NetFlow sharing platform integrated within EmanicsLab. It is assumed that network flow traces collected by each participant are archived on separate storage hosts within their premises. EmanicsLab nodes serve as intermediate switching points for allowing client-to-storage communication with integrated access control. The platform is equipped with

a client-side library offering an Application Programming Interface (API) against which applications can be developed for the purpose of supplying researchers with NetFlow records without the need of attending the searching and retrieval processes. Since it is assumed that the network connection between clients and servers is untrustworthy and unreliable a connection-oriented and cryptographic secured protocol is used for communication. Due to privacy reasons many communities and administrators are reluctant to share exported NetFlow data. A prefix-preserving, cryptography-based, and consistent anonymization algorithm for veiling IP addresses is used in order to overcome these troubles.

The following sections describe this project in more detail.

6.2 Shared Storage Repository for NetFlow Records

IP traffic traces are widely used by researchers for different purposes. On one hand, traces may be used to observe traffic characteristics in order to improve traffic analysis tools. On the other hand they may also be used to evaluate new algorithms based on real traffic traces. One important drawback when designing new algorithms for analyzing IP traffic is the lack of access to real traffic based on which the new algorithm can be evaluated. Often, researchers use traffic traces available locally (such as those captured within a lab network or within a subnet of a university), but they rarely have access to traces collected by other parties.

Within this activity, DATTA aims to design a platform for collecting NetFlow records at multiple sites and make them available to the different project partners. In order to achieve this, the platform needs to offer well-defined interfaces that allow accessing NetFlow records stored locally by the different partners from any host connected to the Internet. Additionally, it shall offer granular access control to these data, as well as the possibility to anonymize the results.

6.3 Design of the Platform

6.3.1 Requirements

The following paragraphs identify the three most important functional requirements:

- **Retrieval of Stored NetFlow Data**. The major objective is to supply researchers with an API for querying flow records from any of the participating partners within their premise without the need of attending the searching and retrieval processes.
- **Access Control**. The platform shall offer an authentication mechanism and granular access control to stored NetFlow data. Each storage provider shall be in the position to independently decide to which users' access on collected flow traces is granted.
- **On-line Anonymization**. Any request for NetFlow records shall result in anonymized records being sent to the requester. The anonymization shall be done on-line and at least mutate the IP addresses stored within the NetFlow records in a prefix-preserving manner.
- **Communication Security**. Since it can basically not be assumed that the communication channels between the distributed components are trustworthy, any communication should be cryptographically secured.

6.3.2 Architectural Overview

The proposed collection and sharing platform features a 3-tiered architecture illustrated by Figure 6-1. A retrieval service running on separate storage nodes (3rd tier), an authentication and forwarding service on EmanicsLab nodes (2nd tier), and a client library offering an API for easily accessing arbitrary NetFlow datasets from any of the available repositories. Secure communication is achieved by a Secure Socket Layer (SSL) that offers client as well server authentication, data integrity, and a cryptographically secured channel for communication. Any query including optional filters and other controlling eventualities results in anonymized NetFlow records being returned to the requester.

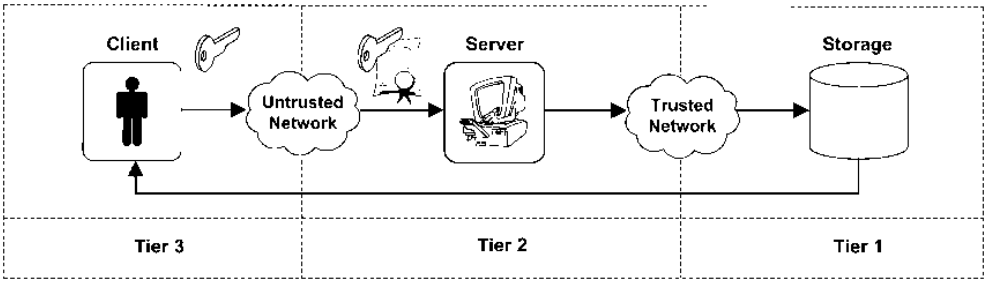


Figure 6-1. 3-tier architecture.

The second tier features a dual property: it acts as a client when requesting flow records from the storage component and at the same time acts as server when handling requests received from remotely residing clients.

Figure 6-2 illustrates the basic information exchange between a client, a server and a storage node and introduces the modular composition of the particular components. The end-user builds applications on top of the client library using the provided API. After successful registration and authentication the client is enabled to request flow data. In a first step a request message needs to be specified containing the location and type of the desired dataset which is then passed to the server by invoking an appropriate remote procedure defined in its interface.

The server receives the client's request message which may be linked to a user-/group-specific anonymization level that is preconfigured by the system administrator at the point in time the user was registered. The server immediately forks a child process in order to be ready for further incoming requests. After logging the request message's content to a file, the message is forwarded to the storage node. The storage service searches the appropriate dataset in the repository, anonymizes all affected source and destination IP addresses, and finally applies the desired filters which are passed within the request message. The remaining dataset is then directly returned to the client as a stream of NetFlow version 5 export datagrams.

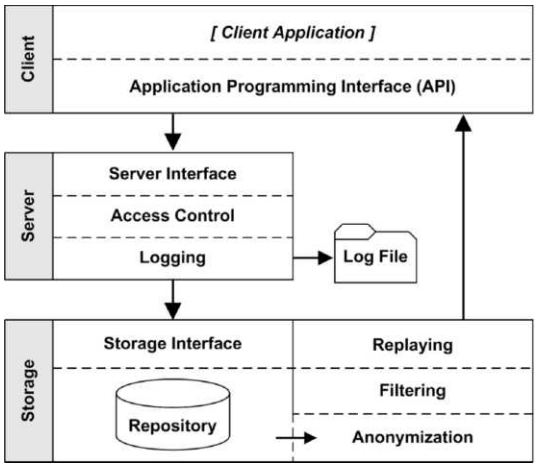


Figure 6-2. Modular component composition.

6.3.3 Components

The system presented here allows any client having a valid and accepted certificate to request offered flow records from any of the participating storage repositories. This section describes the needed components, client, server, and storage, to fulfil the functional requirements specified above.

6.3.3.1 Client Component

The client component serves as an entry point to the sharing network. It enables everybody possessing a valid and accepted certificate to request collected flow traces from provided storage repositories. One of the major objectives is to equip the platform with a client-side library offering a neat and simple API against which applications can be developed for the purpose of supplying researchers with NetFlow records without the need of attending the searching and retrieval processes. Developers shall only have to care about what (i.e. filters) data and from where (i.e. server URL) they require, but not how it is particularly obtained. For large flow traces, the API shall be capable of steadily buffering NetFlow records from a only scarcely controllable UDP stream sent by the storage node's re-play application and simultaneously hand them piecewise out to the application built on top of the provided library. In order to achieve such a controllable retrieval unit a concurrent programming paradigm is used and illustrated by Figure 6-3.

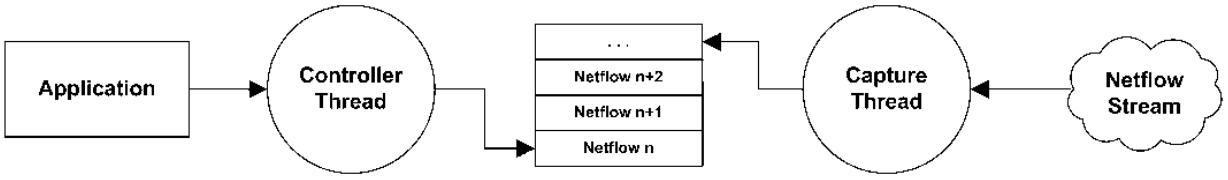


Figure 6-3. A controllable capturing unit.

Incoming NetFlow records are received and decoded by a capture thread that subsequently puts them temporary on a First-in-First-Out (FIFO) organized shared stack. Simultaneously the controller thread pops off the buffered NetFlow records according to user specified events.

Before sending a request to a selected server, the client application binds its receiver unit to an unused local port number and attaches this information to the request message. The port number is used to tell the server's replaying unit where to return the requested data.

The message format used for triggering requests should feature at least the following attributes:

- user identity (i.e. accepted and valid certificate)
- desired server (e.g. emanicslab1.csg.uzh.ch)
- return address (i.e. IP:port)
- requested data (i.e. time-period)
- optional filters (e.g. port- and/or protocol-based)
- delay option (e.g. delay each record by 10 microseconds).

6.3.3.2 Server Component

The server component serves as switching point for client-to-storage communication, performs client authentication, and controls access to provided storage repositories. Each server node maintains a list of X.509 certificates of users to which access is granted. Requesters need to authenticate by using the corresponding private key before their query is forwarded to the storage node. The Public Key Infrastructure (PKI) additionally allows encrypting the communication channel used.

6.3.3.2.1 Server Interface

The server interface allows client applications to communicate with server instances by calling remote methods defined by the interface. The provided interface features the following functionality:

- a function that returns the server's actual state (available/unavailable)
- a function for handing over request messages

Users with a valid and accepted certificate get the necessary privileges for ordering NetFlow datasets by means of at least the following filter options, respectively combinations thereof:

- time-period (i.e. start-/end-time of flow records)
- port numbers (e.g. traffic with destination port > 80)
- protocol types (e.g. TCP, UDP, ICMP)

6.3.3.2.2 Access Control

Everyone interacting with the sharing network needs to authenticate by means of a certificate for the purpose of getting necessary privileges to retrieve flow records of any of the participating storage repositories. In order to get access to server nodes, the client needs to create a self-signed certificate which then is distributed among the server administrators where access is desired. Figure 6-4 illustrates an exemplified scenario.

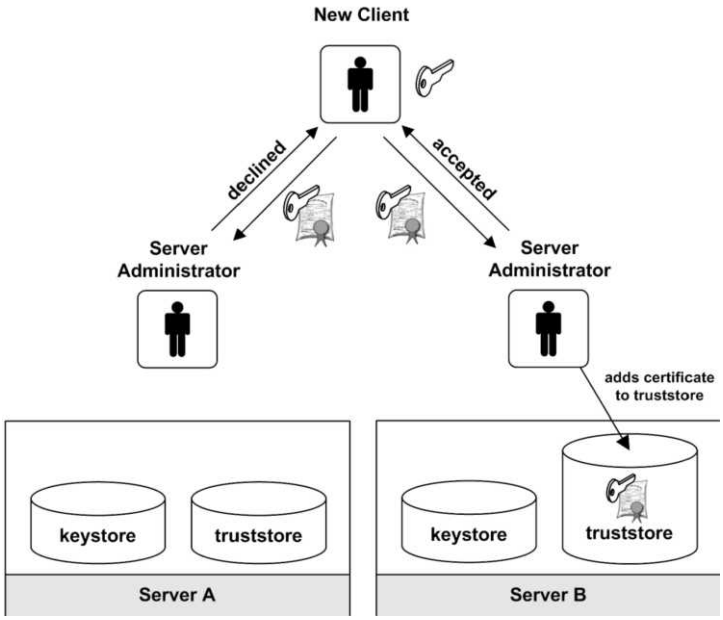


Figure 6-4. Registration process.

6.3.3.2.3 Logging

For the purpose of monitoring the platform's performance and controlling the data exchange, any incoming request respectively, outgoing data stream is logged to a file by the replaying unit at the moment of delivery and should feature at least the following information:

- request timestamp (i.e. YYYY.MM.dd.hh:mm)
- requestor's identity (username, usergroup, IP address)
- applied anonymization options
- time period of requested data items
- filter types applied
- processing time (tin - tout)

6.3.3.3 Storage Component

All data is stored to disk before anyone can request parts of it. This design does not intend to provide functionality to analyze live data. It aims rather to supply interested parties with NetFlow data from the past. It is assumed that the storage repository is already in place and ready to use. For this reason the solution presented here does neither provide any kind of capturing nor collection functionality.

Theoretically, NetFlow records can either be stored in databases or in a file-based fashion. This design supposes that the data is stored as time-sliced files in *nfdump* [55] format. Typically the file names rotate every *n* minutes and rename the output file attaching a time-stamp with the following syntax: *YYYYMMddhhmm*. Assumed that somebody requests flow data from the 18th of March 2008 08:45, the storage unit looks for the file named cap.200803180845. In the case where the requested dataset exceeds the time interval *n*, the affected files are simply concatenated. The highest possible request resolution is not determined by *n*, but rather by the individual flow records stored within such a file.

6.3.3.3.1 Storage Interface

The interface's principal objective is to provide a neat and simple possibility for searching and retrieving sets of flow records from a storage repository independent of the flow record's type and whether it is stored in a database or in a file-based manner. The interface is capable of handling multiple heterogeneous storage repositories attached to the server component like illustrated by Figure 6-5.

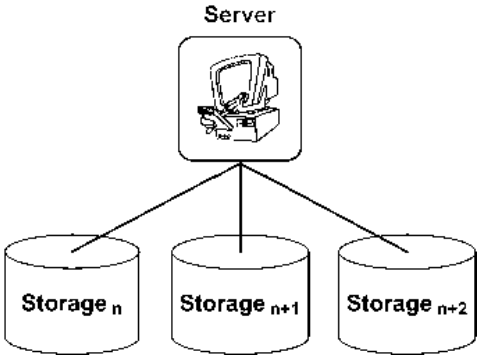


Figure 6-5. Multiple heterogeneous storage repositories attached to a server component.

Due to the fact that NetFlow records come in many different, sometime incompatible formats, the storage platform allows handling more than one specific type of flow record. The interface provided allows its implementation for nfdump flow repositories, flow-tools repositories or any other flow storage solution.

6.3.3.3.2 Anonymization

For the purpose of assuring the user's privacy from which the flow data is captured, a prefix-preserving IP address anonymization algorithm is applied on the shared data. Prefix-preserving means that the original network topology is preserved while anonymization means that critical information parts are removed or at least being veiled. The flow data is stored in its original state and is not being anonymized until it leaves the storage node.

In order to clarify the situation, consider the following scenario: a user of a specific domain where network traffic is captured and analyzed, produces categorical traffic patterns (e.g. sends some mails using SMTP, surfs on specific websites via HTTP, etc.). Assumed that the same user is simultaneously privileged to order these flow records that he, and probably many others in the same domain, has produced - the whole purpose of anonymization is possibly harmed - because he possibly could derive intrinsic IP addresses by analyzing his self-produced traffic patterns.

Since this design only anonymizes IP addresses and the remaining record's information fields, such as port and protocol numbers for example, remain unveiled, the requester possesses already a lot of information about the used anonymization procedure. On top of this comes the fact that prefix-preserving techniques disclose a lot of information about the network topology where the NetFlow data was captured from. Complete anonymization of all the fields of an IP flow record could be performed as well, but in that case the resulting data might not be very useful for some applications that rely on certain characteristics of the flow records. (e.g. analysis of traffic flowing to a web server could not be done anymore).

6.4 Conclusions

The purpose of the work done in this activity was to provide an infrastructure for sharing net-flow traces from many topologically different networks. As a result, researchers and protocol designers have easy access to remotely residing repositories and can use the provided instruments for controlling the ordered NetFlow streams. The use of a Public Key Infrastructure allows providers of such NetFlow repositories to granular control access on the provided data. A well thought-out anonymization algorithm protects the privacy of the users from which the collected and shared datasets were captured.

The implemented prototype is fully functional but its main intention is to serve as basis for further expansions towards a more sophisticated NetFlow sharing platform.

The prototype implementation only adopts the *nfdump* [55] toolkit including a broad scope of its capabilities for processing and replaying stored flow traces. Due to the fact that flow data comes in many different formats the basic implementation should be enhanced by additional flow formats. It seems to be desirable to also adopt *flow-tools* [56] since it is together with *nfdump* a widely used toolkit. Supporting databases have never been a goal for the initial system implementation, but nevertheless the storage interface could easily be enhanced by appropriate solutions for different database instances such as PostgreSQL [57] or MySQL [58] for example.

Since it is assumed that only the communication channel between client and server nodes is untrustworthy (i.e. Internet), the channel between server and storage nodes remains cryptographically unsecured (i.e. local network). However, in the case the application area changes, additional cryptographically secured channels between server and storage nodes may be introduced. Moreover, the way back or more precisely, the NetFlow stream sent from storage nodes to client nodes, is also cryptographically unsecured since existing tools to re-play the NetFlow data from a repository do not support any encryption technology. A way to resolve this restriction would be to decode the collected NetFlow records locally on the storage node and not till then send back to the client node using Remote Method Invocation (RMI) calls via SSL.

7 Policies for Intra & Inter Domain Autonomic Management (PINIDAM)

7.1 Introduction on Policies and Refinement

Policy based approaches to network and systems management are of particular importance because they allow the separation of the rules that govern the behaviour of systems from the functionality provided by that system. This means that it is possible to adapt the behaviour of a system without the need to recode functionality, and changes can be applied without stopping the system. This provides autonomic behaviour in the form of an efficient closed feed-back loop either locally or hierarchically. Research into policy based systems management has focused on languages for specifying policies and architectures for managing and deploying policies in distributed environments. However, policy based techniques are not in widespread use because their advantages in terms of short term return on investment are difficult to justify without significant advances on tools for policy analysis and refinement. In effect, many vendors and system administrators tend to use existing scripting or general purpose programming languages in order to program configuration changes. The benefits of using more

restrictive declarative policy languages such as Ponder, PDL, or CIM-SPL arise only if the policy specification can be analysed for conflicts and inconsistencies, if it is possible to verify that a policy specification preserves well-defined properties and if tools for refining policies from high-level business objectives and Service Level Agreements are available.

Policy refinement is the process of transforming a high-level, abstract policy specification into a low-level, concrete one. Moffett and Sloman [59], identify the main objectives of a policy refinement process as:

- Determine the resources that are needed to satisfy the requirements of the policy.
- Translate high-level policies into operational policies that the system can enforce.
- Verify that the lower level policies actually meet the requirements specified by the high-level policy.

The first of these objectives involves mapping abstract entities defined as part of a high-level policy to concrete objects/devices that make up the underlying system. The second specifies the need to ensure that any policies derived by the refinement process are within terms of operations that are supported by the underlying system. The final objective requires that there is a process for incrementally decomposing abstract requirements into successively more concrete ones, ensuring that at each stage the decomposition is correct and consistent.

Policy based techniques and, in particular, policy refinement are intimately linked to the principles of Autonomic Management. Policy based management can effectively address the issues of Self-configuration and Self-optimization as two of the key properties of Autonomic Systems. These are illustrated in realistic case studies presented in subsequent sections that demonstrate the applicability of the policy refinement techniques described in this work in a realistic application for Quality of Service provisioning and management in IP Networks.

Self-configuration is a key property of Autonomic Systems and is widely accepted that policy based management can address its requirements effectively. Towards that direction, a policy refinement methodology adds an extra layer of automation by simplifying the process of creating enforceable low-level policies. In addition, the system is able to select the optimal method in terms of selected policies, in order to achieve the defined high-level goals. This procedure is in essence Self-Optimisation of the system, as the best available combination of policies is enforced without additional human intervention.

7.2 Goal Elaboration

The policies necessary to control medium-to large-scale systems may be in the order of thousands and must cooperate to fulfil a global system aim. Specifying policies independently is therefore not appropriate and policies must be derived from higher-level administrative goals. Policy refinement is composed of two parts: a) refinement of abstract entities into concrete objects/devices (also known as operationalisation) and, b) refinement of high-level goals into operations, supported by the concrete objects/devices, that when performed will achieve the high-level goal.

Therefore, our motivation is to provide a framework that considers the different aspects involved in the policy refinement process. In order to solve these problems we need a formal representation for objects, their behaviour and organisation, a technique for refining high-level goals into more concrete ones, and finally a means of inferring the combination of operations that will achieve these concrete goals. We use the goal elaboration technique (called KAOS) developed by Darimont et al. [60], [61], to refine high-level goals into concrete ones. However, KAOS is insufficient to achieve our aim because it does not permit to relate systems' behaviour to goal fulfilment. We therefore propose two different techniques in order to provide the formal representation and infer the combination of operations that will achieve the goals. The first technique is based on an Event Calculus formal representation and uses abductive inference for the derivation (see section 7.3) whilst the second technique is based on model-checking (see section 7.4) and uses Linear Temporal Logic as the formalism of choice.

7.2.1 The KAOS Approach

KAOS represents goals as temporal Logic rules and makes use of refinement patterns to decompose these goals into sub-goals that logically entail the original goal. Additionally, this technique makes use of obstacles (negated goals) which are then elaborated and resolved to provide new goals. This process results in a set of refined goals, and the identification of objects and operations that might operationalise them. The final stage of the procedure is to assign each of the refined goals to a specific object/operation so that the final system will meet the original requirements. Whilst the KAOS approach does not provide any automated support for the goal refinement process, it does define a library of domain-specific and domain-independent refinement patterns that have been logically proved correct.

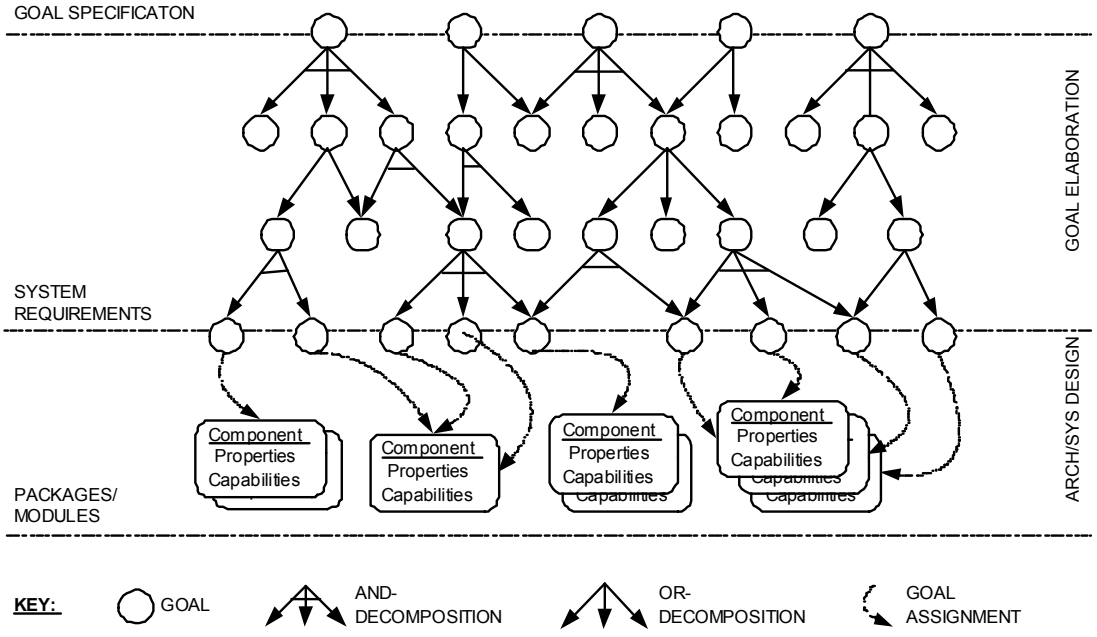


Figure 7-1. Goal decomposition hierarchy.

In KAOS, high-level goals are refined into sub-goals, forming a goal refinement hierarchy where the dependencies between the goals at the different levels of refinement are based on the form of goal decomposition used (AND/OR) or with

temporal relations. Additionally there can be dependencies between goals in different hierarchies. The process of refinement will involve following a particular path down the hierarchy, at each stage verifying the feasibility of achieving the higher-level goal in terms of the lower-level ones. If it is discovered that the goal cannot be achieved, it is necessary to elaborate the information at the higher-level so that suitable lower-level goals can be derived. The decomposition of a goal can be either conjunctive (i.e. only by achieving all the sub-goals can we consider that the higher-level goal is achieved) or disjunctive (i.e. by achieving any one of the sub-goals we can consider the higher-level goal is achieved) or temporally related.

A *domain-independent* goal refinement pattern uses properties of temporal logic operators to provide a proven relationship between a high level goal and a set of sub-goals. For example, the transitivity property of the ◊ R (R will eventually be true) operator provides the following simple domain-independent goal refinement pattern:

$(P \Rightarrow \Diamond R, R \Rightarrow \Diamond Q) \vdash P \Rightarrow \Diamond Q$

If P is true then eventually R is true, AND

If R is true then eventually Q is true, THEN

If P is true then eventually Q is true.

In our example scenario, a domain-specific pattern might be one that describes the sub-goals required to guarantee QoS for a class of application traffic. The user could then refine the goal instance “provide Gold QoS to WebServices applications on the eCommerce Server”, by instantiating this pattern with the Gold class of service and the appropriate application type. Once the user has specified appropriate sub-goals based on the particular pattern, the specification is checked for inconsistencies, e.g. as described in the next section.

Policy based systems use rules to govern their behavioural choices whilst satisfying the goals of the system. Therefore, a policy refinement technique must provide a link between each goal and the underlying system behaviour in order to derive the different ways in which the system can satisfy the goal. This information can then be encoded into policies that control the behaviour of the system as needed. Whilst we can use the KAOS approach to refine abstract goals into lower-level ones, it does not provide a mechanism to connect the goals with the behaviour description of the system.

7.2.2 Policy Refinement Functions and Goal Based Refinement

In an operational environment, policies are refined before and/or during system operation by administrative parties. For this purpose we consider two administrative parties; an Administrator Developer and an Administrator Consultant. The Administrator Developer supports the activities during system design and implementation, whereas the Administrator Consultant carries out the refinement related activities during the start-up and operation of the policy managed system.

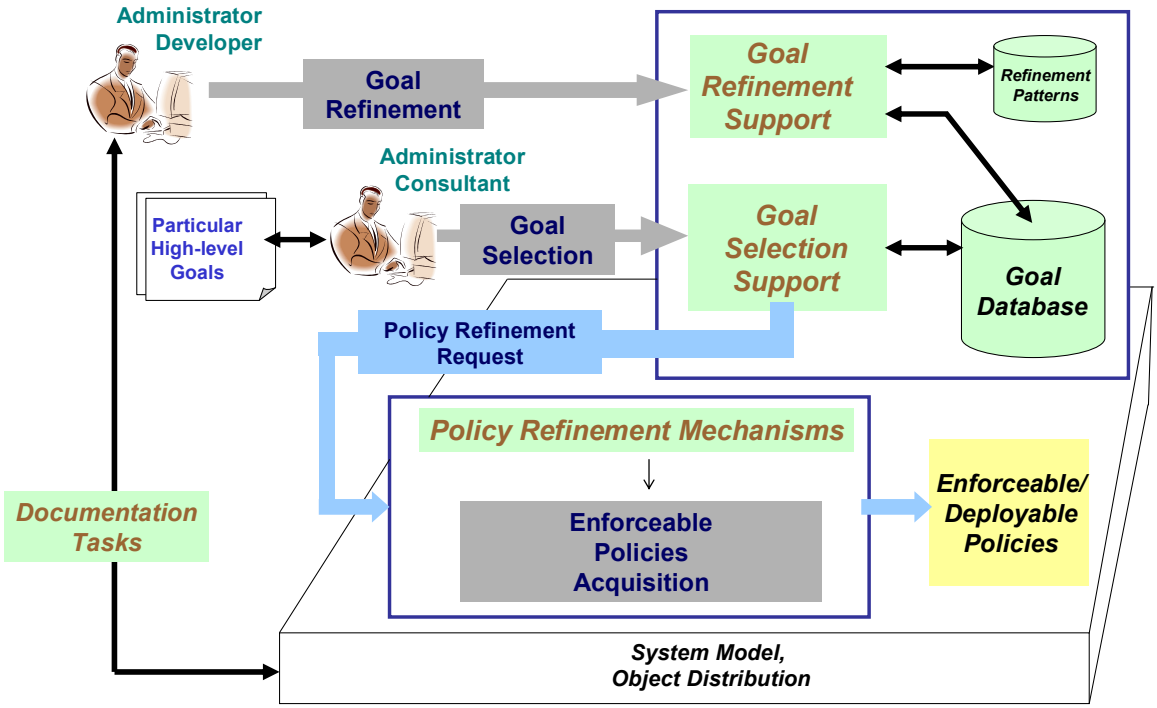


Figure 7-2. Policy refinement functions and goal usage.

As shown in Figure 7-2 the frameworks described in this work provide support for both, the Administrator Developer and the Administrator Consultant in terms of the following activities:

Goal Refinement. This activity is intended to elaborate goal graph structures derived from the High-level Goals that the system must fulfil. These goal-graphs represent both, the business requirements and the different options for achieving them. Refinement is a system-dependent activity and is carried out by the Administrator Developer. Consequently it is carried out during the design and implementation of the system.

Goal Selection. The goal refinement offers many options to decompose a high level goal into sub goals. There are many different ways to go from origin to destination on the goal graph. This activity is meant to allow the Administrator Consultant defining his own option to reach the high level goals. This activity is carried out selecting a given path in the goal graph from the highest level goal to a set of refined sub-goals. We should point out that Goal Selection is carried out at the beginning of the operation of the system operation and at service operation time, if the Consultant is wishing to change his management criteria.

Enforceable Policies Acquisition. This activity is meant to produce a set of system enforceable policies ensuring that goals resulting from the Goal Selection activity will be fulfilled. Acquired policies should include meaningful policy elements such as subjects, targets, events and actions that should correspond to the managed system's capabilities, i.e. refinement produces enforceable policies. In addition, acquired policies should correspond to the managed system's object distribution and deployment. To achieve this so, policy refinement uses a system model including the actual object distribution provided by the Administrator Developer at design/implementation time.

7.2.3 Goal Refinement Support

Figure 7-3 shows the general outline for goal refinement support, which seeks to make use of KAOS [60], [61], to build an application-dependent refinement graph.

The Administrator Developer uses the application-independent refinement patterns together with a model of the managed system's capabilities and supported functions to drive the goal refinement process. Once high-level goals have been defined, these are refined by applying refinement patterns that enable administrator developers to identify lower-level goals. The goal graph structures thus elaborated, are stored in a goal database for eventual use and/or maintenance. This activity should be accomplished during the design and development phase of the system because it is intrinsically related to the system architecture.

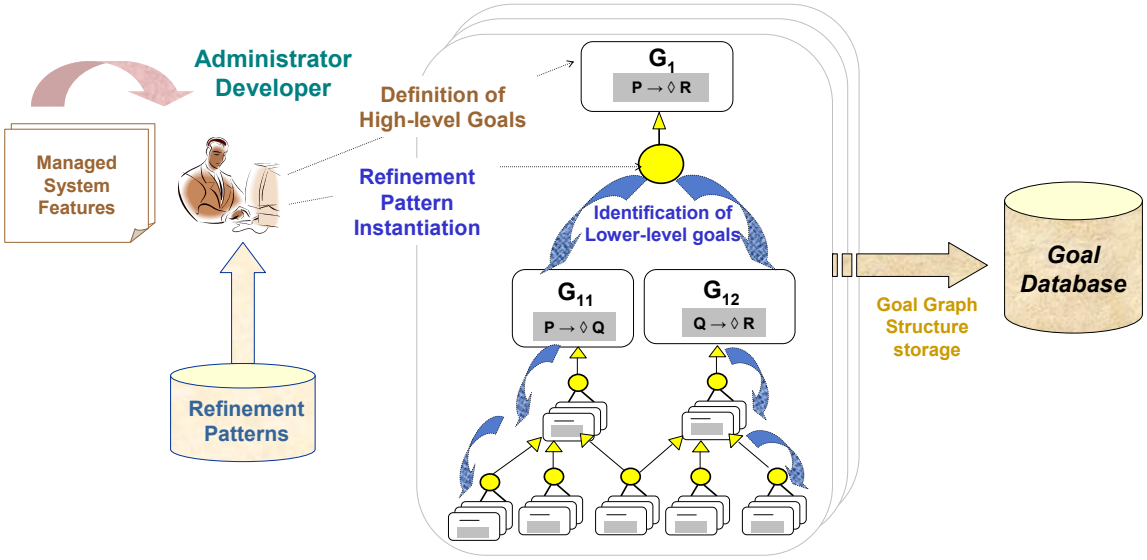


Figure 7-3. Goal refinement usage and elaboration.

In the following sections, in order to illustrate the Goal Refinement function we use an example consisting of a policy-driven Network Dimensioning (ND) system.

7.2.3.1 Identification of Managed System Features

Consider that the developer has identified the following features of the system:

- *General purpose*: The ND sub-system is in charge of assigning network resources according to given traffic predictions in the network.
- *Input*: It receives a traffic estimation matrix that defines the minimum and maximum values for traffic estimations.
- *Output*: It produces configuration sets that may eventually be propagated to enforce the traffic allocation calculations.
- *Capabilities*: It integrates policy based dimensioning mechanisms that influence the allocation of resources and impact the following parameters: hop-counts limit, overall network utilisation, hop-count estimation for delay and packet loss estimations, explicit allocations, distribution/reduction of extra link capacity.

7.2.3.2 Defining the Nature of High-level Goals

Administrators must ensure a given system behaviour and assign network resources according to specific traffic distributions. In essence, this constitutes high-level goal (G1), named as G1 *ConfigurationDirectivesSet* and expressed in natural language as *"when a network dimensioning request is received, the configuration directives should be generated and eventually stored and propagated to the underlying components"*. Other high-level goals could be defined for the ND system for which the intersection of all would become the different targets of interest that the Administrator Developer has envisaged as controllable for the ND system.

Goal G1 can be formalized in Linear Temporal Logic as follows:

G1 ConfigurationDirectivesSet: ND_Request$\rightarrow\Diamond$DirectConfig&Propagated

7.2.3.3 Goal Refinement Patterns

Once high-level goals like G1 have been identified, they are decomposed into sub-goals. KAOS prescribes to tackle this issue by asking HOW questions, i.e. in the ND example to determine how the system can configure dimensioning directives. The ultimate objective is to achieve some behaviour in the system. For this purpose, different propositional patterns would be useful. For example, consider the use of KAOS patterns to refine the application-independent *Achieve* goal $P \rightarrow \Diamond Q$ shown in the table below.

ID	Formal Representation of sub-goals	Description
RP3	$P \rightarrow \Diamond R$ $R \rightarrow \Diamond Q$	Milestone-driven refinement
RP4	$P \wedge P1 \rightarrow \Diamond Q1$ $P \wedge P2 \rightarrow \Diamond Q2$ $\Box(P1 \vee P2)$ $Q1 \vee Q2 \rightarrow Q$	Case-driven refinement
RP5	$P \wedge \neg R \rightarrow \Diamond R$ $P \wedge R \rightarrow \Diamond Q$ $P \rightarrow \Box P$	Conditional milestone-driven
RP6	$\neg R \rightarrow \Diamond R$ $P \wedge R \rightarrow \Diamond Q$ $P \rightarrow \Box P$	Unconditional milestone-driven

Table 7-1. Different patterns to refine the Achieve goal $P \rightarrow \Diamond Q$.

The use of one pattern instead of another depends on the system's capabilities and the developer's criteria. In our example, suppose that P represents an *ND_Request* - reception of a network dimensioning request and Q represents *DirectConfig&Propagated* - ND directives generated and propagated. Suppose also that R represents *TM_reception* reception of the traffic matrix. The patterns shown in the table above can be applied as follows:

- The **Milestone-driven refinement pattern (RP3)** proposes that *"on the occurrence of a state P, an intermediate state R must first be reached from which a goal state Q must eventually be reached"*.
- The **Case-driven refinement pattern (RP4)** proposes that *"on the occurrence of P, an alternative sate P1 that would satisfy Q1 or P2 satisfying Q2 would be sufficient to satisfy the state Q".*

- The **Conditional milestone-driven pattern (RP5)** proposes that *"on the occurrence of P and under the absence of the state R, it is a requirement that state R and state P both hold so that the goal state Q eventually holds". "In addition, state P should hold "always" during the execution of the system*.
- The **Unconditional milestone-driven pattern (RP6)** proposes that "*on the absence of a state satisfying R, the latter must hold when P occurs so that the goal state Q can be eventually reached"*.

Continuing in the ND example, the developer evaluates how the above four alternatives can match the capabilities and the operation of the ND system to decide which refinement pattern is better suited to use to decompose the *G1 ConfigurationDirectivesSet* goal.

For instance, if we associate P to *ND_Request*, Q to *DirectConfig&Propagated* and R to *TM_reception*, then RP5 is discarded because the reception of a configuration request (*ND_Request*) is a discrete event that cannot happen continuously until the traffic matrix (*TM_reception*) is received nor until the directives are configured and propagated (*DirectConfig&Propagated*). RP6 is also discarded because the reception of a traffic matrix (*TM_reception*) is also a discrete event that cannot hold until the *ND_Request* state is reached.

The two remaining options are then a milestone-driven tactic (refinement pattern RP3) and a refinement by cases (refinement pattern RP4). Let us assume that the dimensioning of the network can be achieved in two ways: a) considering the allocation of resources to cope with the minimum demand only, or b) considering the allocation of minimum demand plus the allocation of the remaining network resources. These two options are mutually exclusive and both can be used to configure the directives in the ND system. For this reason, the milestone-driven pattern RP3 is discarded because it doesn't allow these alternatives

Finally, bringing the case-driven refinement pattern RP4 into our Network Dimensioning example formalises two different alternatives of how to refine the original G1 ConfigurationDirectivesSet goal. More concretely, the developer brings the case-driven RP4 pattern to refine G1 as he considers that the High-level Goal G1 could be fulfilled in two ways which in turn define the goals G2 and G3 respectively: G2 opts to dimension the network considering the allocation of minimum demand; G3 opts for network dimensioning considering the allocation of minimum demand plus the usage of the remaining network resources. These two refinements are graphically represented in the upper part of Figure 7-4 and are formally expressed as follows:

G1 ConfigurationDirectivesSet: ND_Request → ◊ DirectConfig&Propagated

G2 MinDemandStrategy: ND_Request /\ minDemandStrategy → ◊ DirectConfig&Propagated

G3 MinWExtraCapStrategy: ND_Request /\ minWExtraCapStrategy → ◊DirectConfig&Propagated

We must acknowledge that the Goal Refinement process is aimed at elaborating goal graph structures of High-level Goals that the system can handle. These represent both, the requirements and the different options with which the High-level Goals could be achieved. As described later, one of the above two options to generate the configuration directives would be selected by means of the Goal Selection process during the start up and/or the operation of the system.

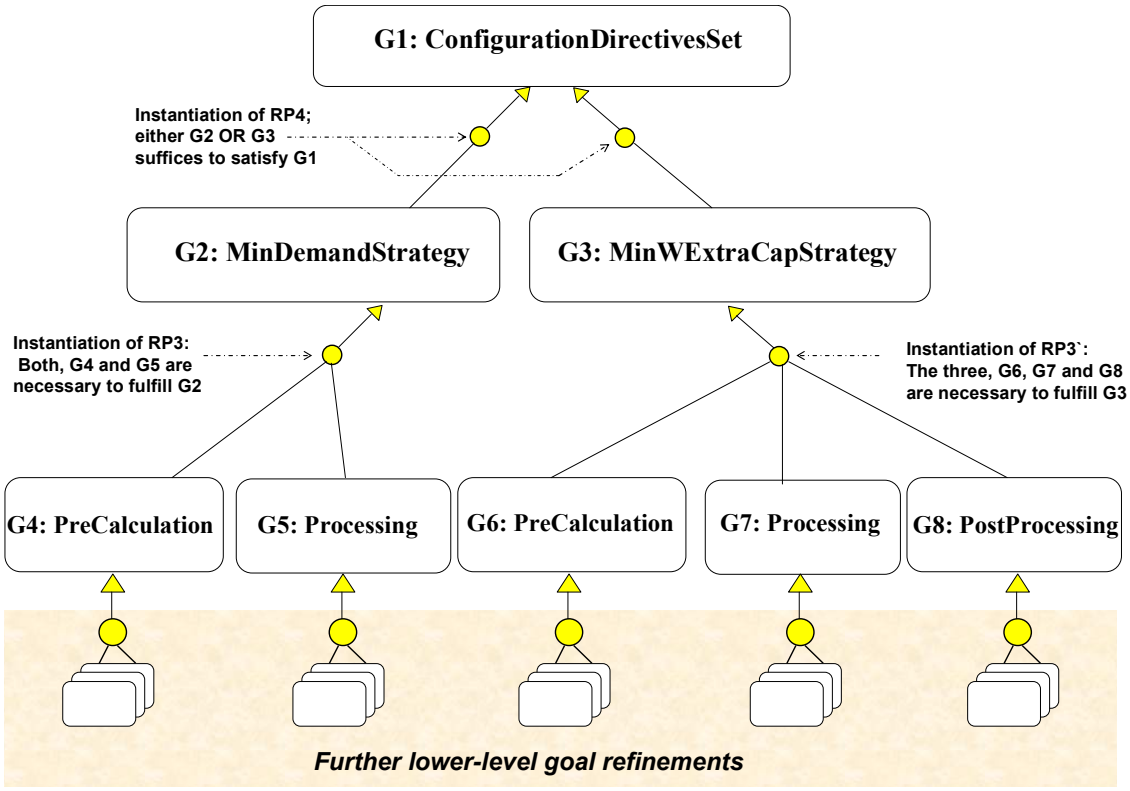


Figure 7-4. Initial goal graph structure elaboration process.

The patterns shown in Table 7-1 can be extended to make them suitable for use in particular application domains. For instance, RP3 and RP4 could be extended to deal with multiple milestone-driven and multiple case-driven situations as shown in Table 7-2.

RP	Formal Representation of sub-goals	Description
RP3'	$P \rightarrow \Diamond R \quad R \rightarrow \Diamond S \quad S \rightarrow \Diamond Q$	Multiple milestone-driven refinement
RP4'	$P \wedge P1 \rightarrow \Diamond Q1, \; P \wedge P2 \rightarrow \Diamond Q2, \; P \wedge P3 \rightarrow \Diamond Q3, \; \Box(P1 \vee P2 \vee P3), \; Q1 \vee Q2 \vee Q3 \rightarrow Q$	Multiple case-driven refinement

Table 7-2. Extension of the refinement patterns RP3 and RP4.

Going further in the refinement process of our example, G2 MinDemandStrategy and G3 MinWExtraCapStrategy should be further refined. For G2 MinDemandStrategy, the developer considers that this requirement must be accomplished in two steps: a) a pre-calculation of resources and b) a processing step that should take into account the pre-calculations. Consequently, the goal G2 MinDemandStrategy is refined into the sub-goals G4 PreCalculation and G5 Processing by bringing the pattern RP3 into the context of the refinement of G2 (see Figure 7-4) This goal sub-tree formalises the requirement of achieving network dimensioning configurations in two steps; a pre-calculation AND the processing step.

Similarly, for the goal G3 MinWExtraCapStrategy, the developer considers necessary three sub-processes, namely PreCalculation, Processing and PostProcessing. Consequently this goal is refined into the sub-goals G6 PreCalculation, G7 Processing

and G8 PostProcessing, by the bringing the extended pattern RP3' into the context of the refinement of G3 (see Figure 7-4)

A similar procedure has to be carried out to refine each lower-level goal (G4 to G8). For example, the PreCalculation sub-goal (G4 and G6 in Figure 7-4) is refined into two alternative cases: one that considers delay/loss estimations (G9 delayLossEstimated), and another that considers delay/loss estimations with explicit resources allocation (G10 delayLossEstimationWithExplicitAllocation). The result is a goal graph structure linked to the PreCalculation goal as shown in Figure 7-5. Further refinements in each branch are needed until it is feasible to relate the refinements with specific system states. In fact, KAOS considers that goals identifiable with a system state can be considered as a lowest-level goal. For example, this situation occurs here with the refinement applied to the G9 delayLossEstimated sub-goal.

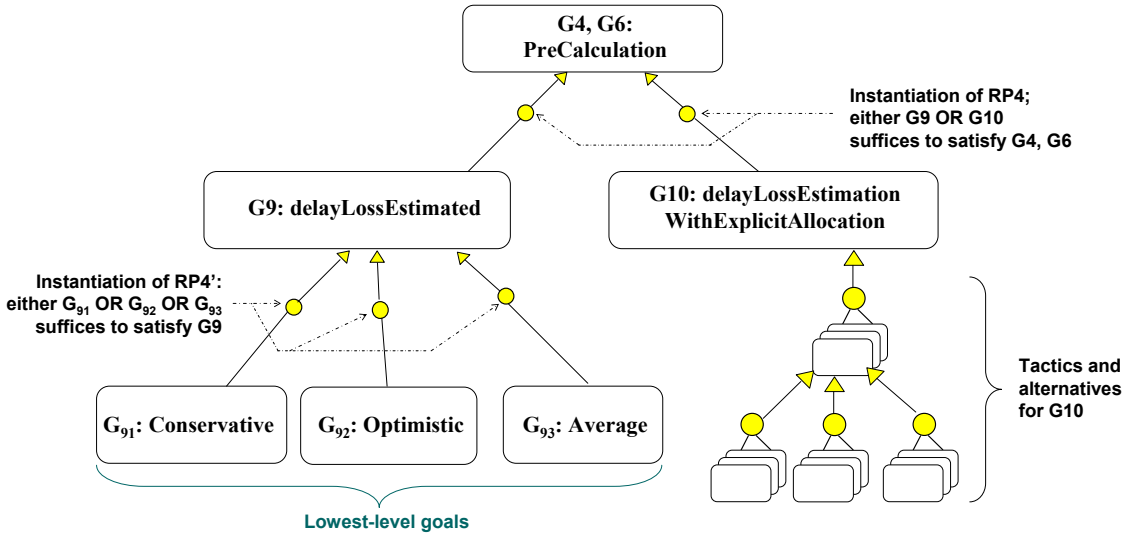


Figure 7-5. Goal graph sub-tree for the precalculation sub-goal.

The sub-goal G9 is further refined into three alternative cases by means of the extended refinement pattern RP4'. The result is a set of three refinements expressed as G_{91} Conservative, G_{92} Optimistic and G_{93} Average. G_{91} considers the maximum delay and the maximum packet-loss recorded in the network as the basis to estimate the delay and packet loss during the dimensioning process; taking the maximum values for each link is considered to be a *conservative tactic* in the pre-calculation process. On the other hand, G_{92} considers the minimum values for delay and packet loss; this is considered an *optimistic tactic*. Finally G_{93} proposes a tactic that considers the *average values* of delay and packet losses in every link of the network, as the base to estimate the delay and packet loss during the calculation process. These three goal refinements (G_{91}, G_{92} and G_{93}) are specific enough to relate them to specific system states and are considered as the lowest-level goals that cannot be further refined.

7.2.4 Goal Selection Support

Once system goals refinement is completed, the goal graph structures should be stored in databases for further use and maintenance. Having in mind that goal graph structures formalise potential requirements and options to fulfill High-level Goals, they will be used to select a concrete one among the different available alternatives.

Goal Selection Support consists of browsing through the previously elaborated goal graph structures stored in the goal database and selecting the strategies that better reflect the consultant administrative criteria.

In our example, consider the case where the consultant wants to provide configuration directives only for the minimum demand estimations. Consider also the situation where the consultant wants to have better chances to fulfil the SLAs of real-time services. This is his particular administrative guideline to navigate through the graph that finally yields to the selection of the path reflected in Figure 7-6 with shadow boxes.

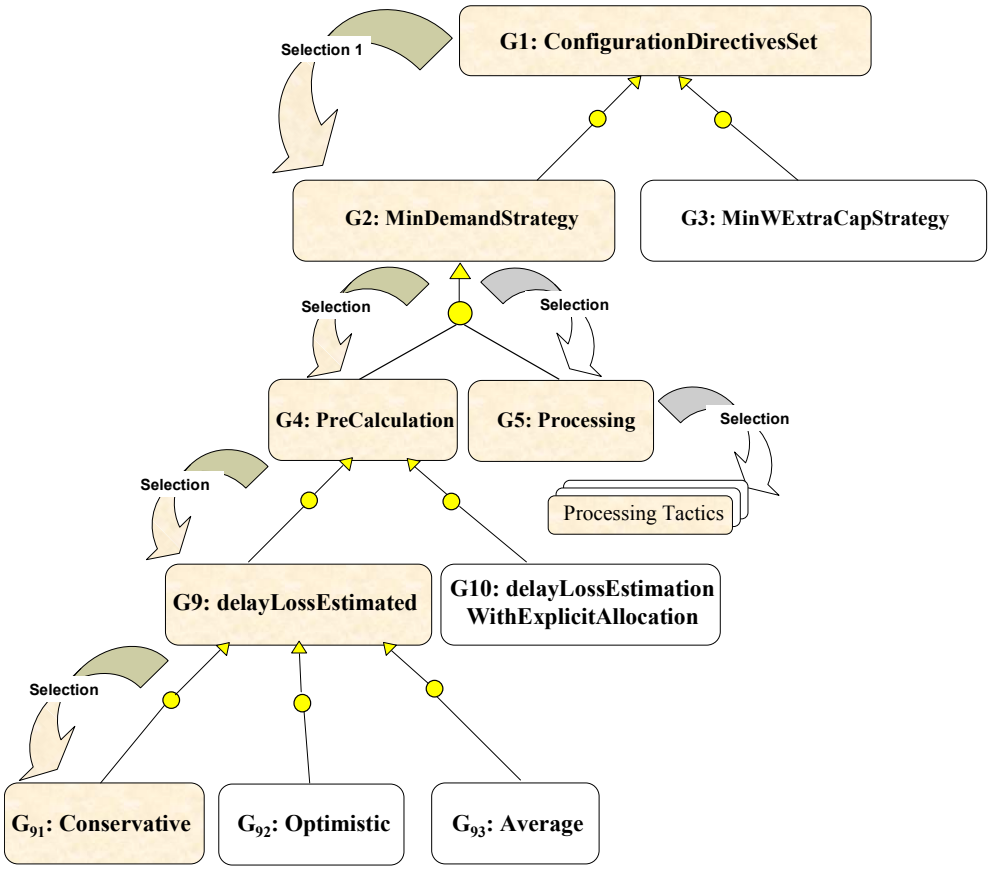


Figure 7-6. Sample of a goal selection action.

7.2.4.1 Policy Refinement Mechanisms

The Policy Refinement Mechanisms support the *Enforceable Policies Acquisition* activity of the policy refinement framework of Figure 7-2. The target of these mechanisms is to determine policies fulfilling the goals resulting from the Goal Selection. With this aim, we make use of the logical foundations of the goal elaboration methodology, reactive systems analysis techniques and novel concepts developed on purpose to make the policy refinement activity a systematic process. In this sub-section we first provide a general outline of the policy refinement mechanisms and then we describe the involved sub-processes.

7.2.4.1.1 General Outline of the Policy Refinement Mechanisms

The principle of the policy refinement mechanisms is to abstract enforceable policies that can be deployed onto the managed system to commit with a given goals selection. The whole process can be decomposed into four sequential steps as follows: a) *Establishment of Temporal Relationships*, b) *Enforcement of System Behaviour*; c)

Translation Process, and d) *Encoding of Deployable policies*. These processes are graphically represented in Figure 7-7 and are briefly outlined hereafter.

1. *Establishment of Temporal Relationships*. This process is aimed at characterising the lowest-level goals as system specifications that are suitable for use with automated analysis techniques.
2. *Enforcement of System Behaviour*. This process is aimed at forcing the underlying managed system behaviour so that the specifications provided by the "Establishment of Temporal Relationships" process can be fulfilled.
3. *Translation Process*. This process is aimed at identifying a subset of transitions in the restricted system behaviour as event-condition-action policies.
4. *Encoding of Deployable policies*. This process produces the policies that should be deployed onto the actual managed system from the policy-controlled transitions provided in the previous step.

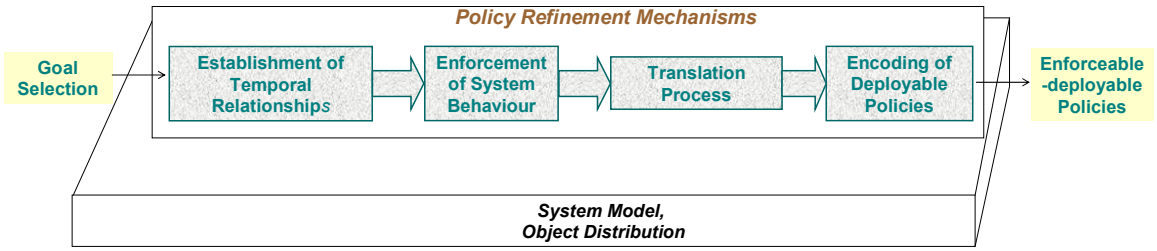


Figure 7-7. Policy refinement mechanisms.

7.2.4.1.2 Establishment of Temporal Relationships

The main activities behind this process can be summarised as: a) identification of temporal relationships between refined goals (lowest-level goals), and b) representation of these temporal relationships by means of suitable formats for use with automated analysis techniques. This approach is similar to the principle of using temporal logics to express search control knowledge for planning techniques [62]. The following is the rationale of these two activities.

For the identification of temporal relationships between refined goals, we use the fact that the selected goals are a sub-set of KAOS goal graph structures and as such, they are intrinsically time-related. Consider for example a parent goal G_1 refined into G_{11} and G_{12} according to the refinement pattern RP3 as shown in Figure 7-8. For this refinement, KAOS establishes the temporal prescriptions of the parent goal (tp_1) and of its corresponding refinements (tp_{11} and tp_{12}) as follows:

- tp_1, formally expressed as $P \rightarrow \Diamond Q$, identifies that *"under the occurrence of a state P, the state Q must eventually be reached"*.
- tp_{11}, formally expressed as $P \rightarrow \Diamond R$, identifies that *"under the occurrence of a state P, the state R must eventually be reached".*
- tp_{12}, formally expressed as $R \rightarrow \Diamond Q$, identifies that *"under the occurrence of a state R, the state Q must eventually be reached".*

Central to our study are the Temporal Relationships (TR) between the goal refinements: in this particular example between G_{11} and G_{12}. In a strict temporal ordering of properties, the temporal prescriptions of these refinements (tp_{11} and tp_{12}) suggest that property P should hold before property R from which property Q should eventually hold.

In other words, this ordering of properties implies that G_{12} should be fulfilled after G_{11}, which establishes a temporal relationship between these two goal refinements. Figure 7-8 shows the Temporal Relationship TR_1 which formally states that *"on the occurrence of a state P, an intermediate state R must first be reached from which a goal state Q will eventually be reached"*.

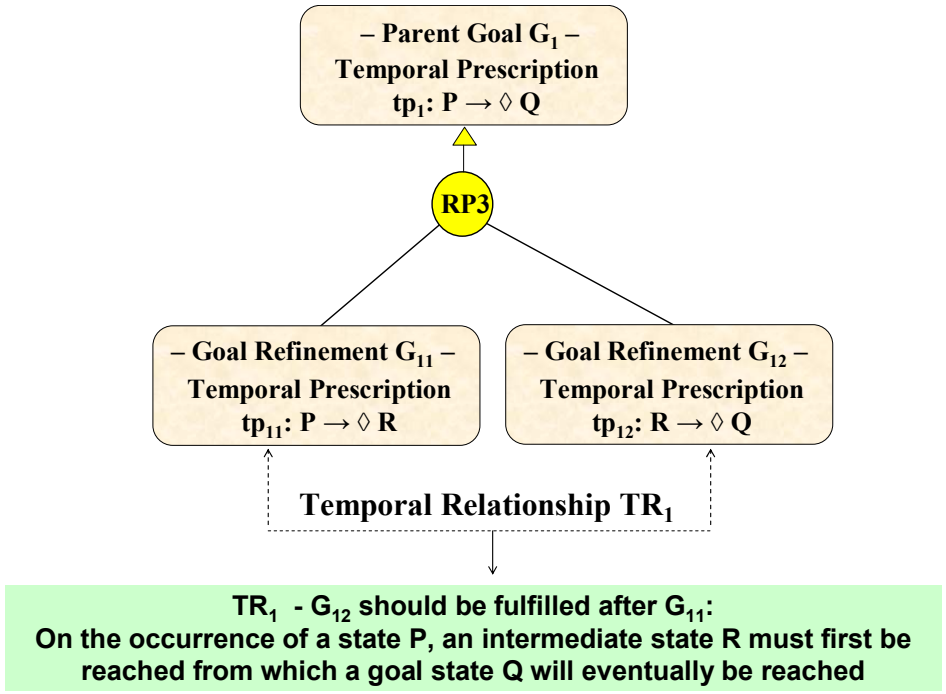


Figure 7-8. Temporal relationships of goal refinements.

Having identified the temporal relationships between goal refinements, the second key aspect is the characterisation of these temporal relationships with formal representations that are suitable for use with automated analysis techniques. In other words, we need a formalism to express the ordering of events in time. To this aim we have followed the principles of Finite-state specification patterns [63]. It is worth noticing that these patterns are different from the patterns used to refine goals. The Finite-state specification patterns are used to specify behavioural properties of reactive systems.

Finite-state specification patterns are classified in two main groups [63]: Occurrence Patterns and Order Patterns. While Occurrence Patterns are used to identify behaviours in which a specific state/event takes place, Order Patterns deal with prescribed behavioural arrangement of states/events. A complete classification of these patterns is shown in the upper-left part of Figure 7-9.

Occurrence Patterns are classified in *Existence, Absence, Universality* and *Bounded Existence*. *Existence* patterns should be used in cases where the most important is to specify that a state/event occurs. *Absence* patterns should be used when it is necessary to specify that a state/event does not occur. *Universality* patterns are meant to specify that states/events occur always throughout a given scope. *Bounded Existence* patterns should be used in situations where it is important to specify the number of times a state/event occurs within a scope. *Order Patterns* are classified in *Response* and *Precedence*. *Response* patterns are used to represent constraints in the order of states/events. *Precedence* patterns are concerned with the specification that a given state/event *P* must be always preceded by a state/event *Q* within a scope.

In this context, a scope represents a restriction on the time a given property must hold. For example, with regard to an Existence Pattern, a scope would be used to define whether the occurrence of a state/event holds either globally throughout the execution of the system, or before, or after or between other situation(s). The literature [63] has identified five possible scopes: *Global*, *Before*, *After*, *Between* and *After-until.* A graphical representation of these scopes is shown in the upper-right part of Figure 7-9.

A Global scope prescribes that a pattern holds for the entire system execution; A Before scope is used to indicate that a pattern holds throughout the execution of the system up to a given state/event. The After scope is used in situations where the pattern must hold after a given state/event and throughout the execution of the system. The Between scope helps define behavioural situations in which a pattern must hold at any part of the execution of the system from a given state/event to another state/event. Finally the scope After-until is like Between with the difference that in the former the designated part of the execution continues even if the second state/event does not occur.

The study of the above patterns, scopes and their representation in different logics has been the subject of research for some time [63].Thousands of combinations of pattern/scopes have been identified in the literature and practical approaches have been proposed to elaborate databases that classify combinations of patterns/scopes [64]. The classification of pattern/scopes with their corresponding logical representations enables to find the formal representation of practically any requirement in a systematic manner. The following presents how these concepts have been laid down in our approach.

Consider G_{11} and G_{12} from our previous Goal Selection example and two temporal relationships as follows:

- TR1: "G_{12} is always fulfilled after G_{11}"
- TR2: "G_{12} is never fulfilled after G_{11}"

Figure 7-9 illustrates the process of specifying TR1 and TR2 by making use of the pattern/scope approach described above. By means of behavioural patterns/scopes we could express such temporal relationships as formal specifications of Linear Temporal Logic (LTL) in a systematic manner [64]. As in previous sections we use the classical temporal operators: $\Diamond$ eventually in the future, $\Box$ always in the future, and the classical logic connectors $\wedge$ and, $\vee$ or, $\neg$ not, $\rightarrow$ logical implication, $\leftrightarrow$ equivalence, and so forth.

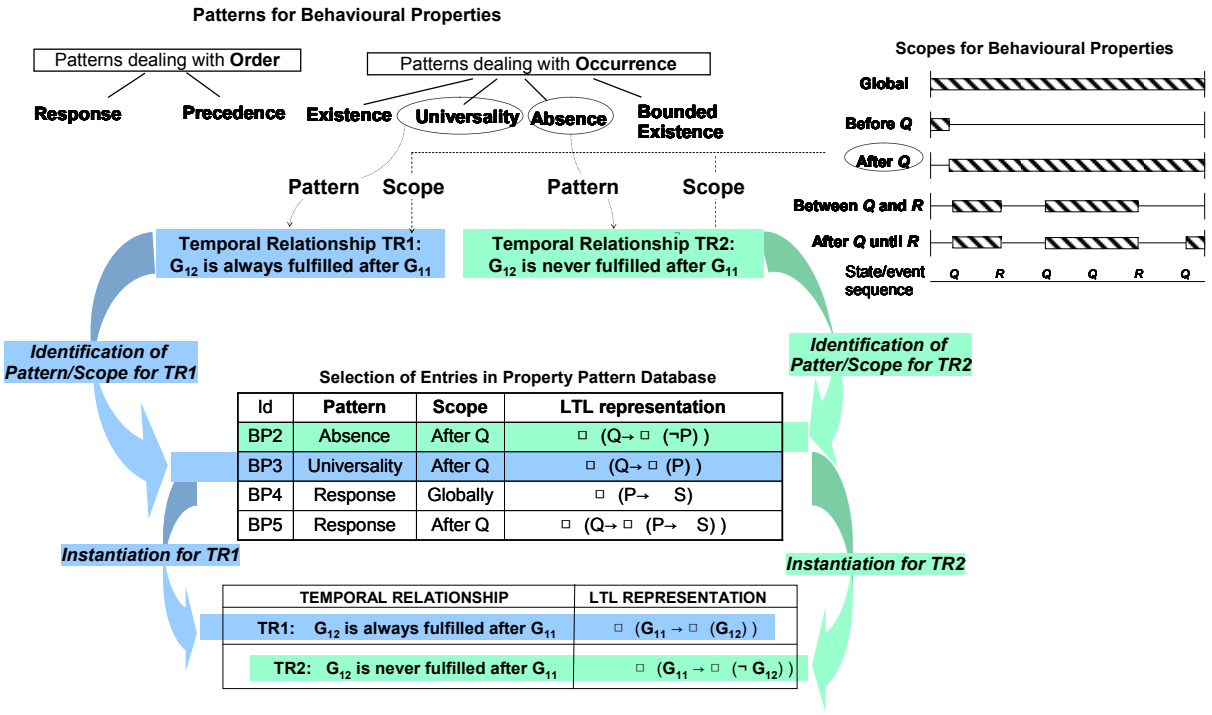


Figure 7-9. Formalisation of temporal relationships.

The key issue is to define which combination of pattern/scope to retrieve from the database of pattern/scopes. Pattern/scopes databases contain entries of pattern/scopes combinations as shown in the *Selection of Entries in Property Pattern Database* in the central part of Figure 7-9.

- For TR1 we could consider that G_{12} always holds after the fulfilment of G_{11}. This temporal relationship fits into the prescription of a Universally/After Pattern/Scope combination and then we instantiate BP3 to represent TR1 in LTL notations. The LTL representation of TR1 is as follows: $\Box(G_{11} \rightarrow \Box(G_{12}))$
- For TR2 we could consider that G_{12} is always absent after the fulfilment of G_{11}. This temporal relationship fits into the prescription of an Absence/After Pattern/Scope combination and then we instantiate BP2 to represent TR2 in LTL notations. TR2 is represented as follows in LTL: $\Box(G_{11} \rightarrow \Box(\neg\ G_{12}))$.

Temporal relationships between goals not belonging to the same branch can be discovered going up into the graph hierarchy. For example, consider the composite Goal Selection shown in Figure 7-10. Given that G_{11} and G_{12} are not lowest-level goals, the fulfilment of the highest-level goal G_1 should consider not only the temporal relationship TR_1 but also the temporal relationship of the lowest-level goals, TR_{11} and TR_{12}. Consider that the temporal relationship TR_1 prescribes that "G_{12} should be fulfilled after G_{11}"; that TR_{11} prescribes that "G_{112} should be fulfilled after G_{111}"; and that TR_{12} prescribes that "G_{122} should be fulfilled after G_{121}". Under these circumstances, the temporal relationship that characterises the fulfilment of G_1 may prescribe that the lowest-level goals should be fulfilled in the following ordering: G_{111}, G_{112}, G_{121} and G_{122} for which a similar approach may be followed to characterise this temporal relationship with formal LTL representations.

In our approach, by using the KAOS methodology, we establish a hierarchy amongst goals, formalised through goal graph structures. This concept has been crucial to consider the fulfilment of High-level Goals as the combination of the fulfilment of lowest-

level goals. Given that lowest-level goals are connected or linked through temporal relationships, we can view the fulfilment of High-level Goals as the fulfilment of time sequences of lowest-level goals. This KAOS-based approach is different to traditional plan-based techniques in which a goal is identified by mere state predicates [65] or other traditional approaches in which goals have been acknowledged as a set of desirable final states [66]. This approach takes advantage of temporal relationships to carry out automated analysis techniques.

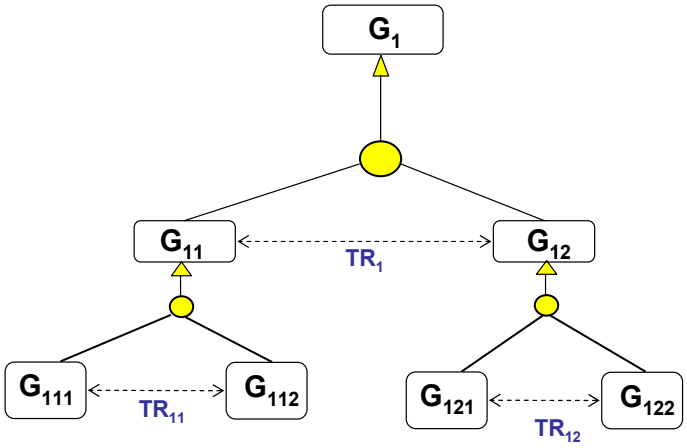


Figure 7-10. Temporal relationships for composite goal selections.

Whilst the methodology described above permits to elaborate goals into composite goal selections it is still necessary to be able to automatically derive the policies that achieve these goals. To this end we present two techniques that have been developed that are based on different formal models and derivation techniques namely: model-checking and Abductive Reasoning (itself part of the wider class of logic programming based techniques). Both approaches are illustrated based on network QoS management scenarios derived from the TEQUILA project network management platform (www.ist-tequila.org).

7.3 Logic-Programming Based Techniques for Policy Refinement

At a given level of abstraction there will be some description of the system (SD) and the goals (G) to be achieved by the system. The relationship between the system description and the goals is the Strategy (S), i.e. the Strategy describes the mechanism by which the system represented by SD achieves the goals denoted by G. Formally this would be stated as:

$$SD_x, S_x \vdash G_x \quad \text{where x is a label denoting the abstraction level} \qquad (1)$$

So, in this approach, a representation of the system description, in terms of the properties and behaviour of the components is necessary. The behaviour of the system is defined in terms of the pre- and post-conditions of the operations supported by the components, which the user can specify using a high-level notation such as state charts. Since the goals to be satisfied can be defined in terms of desired system states, they can be specified in a notation similar to that used to specify the post-conditions of the operations.

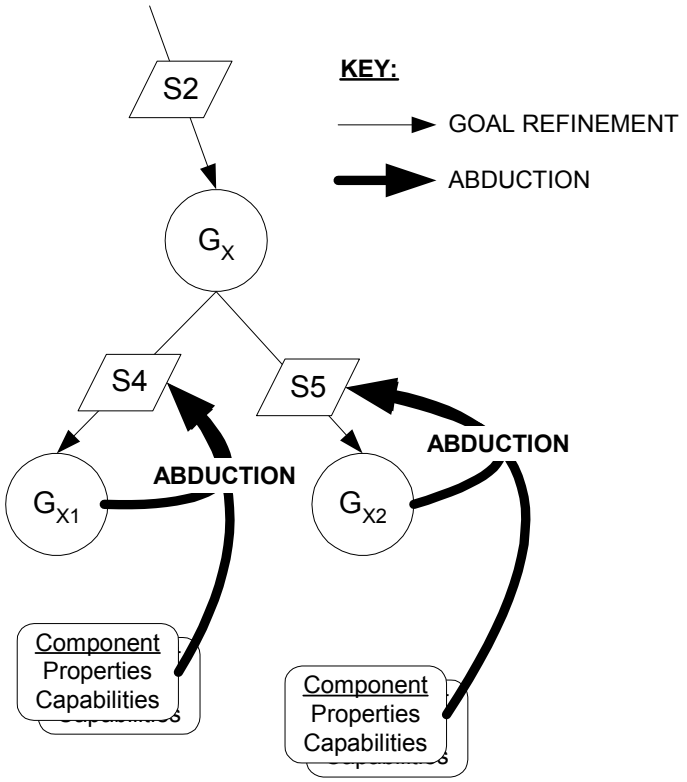


Figure 7-11. Deriving strategies from goals and system description.

The system description information is then transformed into a formal representation that supports automated analysis. Given the relationship between the system description, strategy and goal defined in (1) above we then use abduction to programmatically infer the strategies that will achieve a particular goal (Figure 7-11). Additionally, we can use the properties of the goal decomposition approach described previously to decompose the system description and strategies as follows:

$$G_{x1}, G_{x2}, \ldots, G_{xN} \vdash G_x \quad \text{goal decomposition} \qquad (2)$$

$$\begin{aligned} &SD_{x1}, S_{x1} \vdash G_{x1} \\ &SD_{x2}, S_{x2} \vdash G_{x2} \quad \ldots \\ &SD_{xN}, S_{xN} \vdash G_{xN} \quad (\text{from } 1) \end{aligned} \qquad (3)$$

This shows that if there is some combination of lower level goals from which we can infer the original goal, then for each of these sub-goals there must be a corresponding strategy and system description combination which will achieve it. Therefore, provided the goal decomposition is correct, intuitively the combination of the lower level system descriptions should allow inference of the abstract system description and similarly the combination of the lower level strategies should allow inference of the abstract strategy.

As mentioned previously, the other component of the refinement process is to refine abstract entities into concrete objects/devices in the system. For example, in the system illustrated in Figure 7-1, there might be an abstract entity called “Network” that logically consists of the “Engineering Network”, “Core Network” etc., where each of these in turn consist of the routers and servers within them. We propose that a domain hierarchy be used to represent the relationships between the various abstract entities and the low-level concrete objects [67].

Domains provide a means of grouping objects to which policies apply and can be used to partition the objects in large systems according to geographical boundaries, object type, responsibility and authority. Membership of a domain is explicit and not defined in terms of a predicate on object attributes. An advantage of specifying policy scope in terms of domains is that objects can be added and removed from the domains to which policies apply without having to change the policies. The formal representation of the domain structure is as shown in [68].

In order to implement the approach outlined above, it is necessary to have a formal representation of the system description; and the strategies and goals. However, for the implementation to be usable, it would be ideal to be able to model the systems in a high-level notation and translate this into Event Calculus for analysis purposes. UML would be well suited for this purpose since it is widely used and is supported by many commercial tools.

This rest of this section outlines how UML would represent each of the types of information that need to be modelled together and describes how they can be translated into Event Calculus. The formal language being used is based on that described in [68], where in addition to the base predicates and axioms of Event Calculus we make use of the function symbols shown in Table 7-3 below.

7.3.1 System Description

The system description models the objects in the system in terms of their behaviour. Using the following notation, it is possible to illustrate the use of these rules for modelling system behaviour. So, let us say there is an object of type `DiffServRouter` in the example system. This type has attributes to hold the IP interfaces and actions to configure various parameters of the router, which might be represented in UML as a class diagram. The actions for the `DiffServRouter` type can be specified in a UML state chart representation as shown in Figure 7-12.

Symbol	Description
`state(Obj, V`$_O$`, Value)`	Represents the value of a variable of an object in the system. It can be used in an `initiallyTrue` predicate to specify the initial state of the system and also as part of rules that define the effect of actions.
`op(Obj, Action(V`$_P$`))`	Used to denote the operations specified in an action event (see below)
`systemEvent(Event)`	Represents any event that is generated by the system at runtime. The Event argument specified in this term can be any application specific predicate or function symbol.
`doAction(ObjSubj, op(ObjTarg, Action(V`$_P$`)))`	Represents the event of the action specified in the operation term being performed by the subject, `ObjSubj`, on the target object, `ObjTarg`.

Table 7-3. Function symbols.

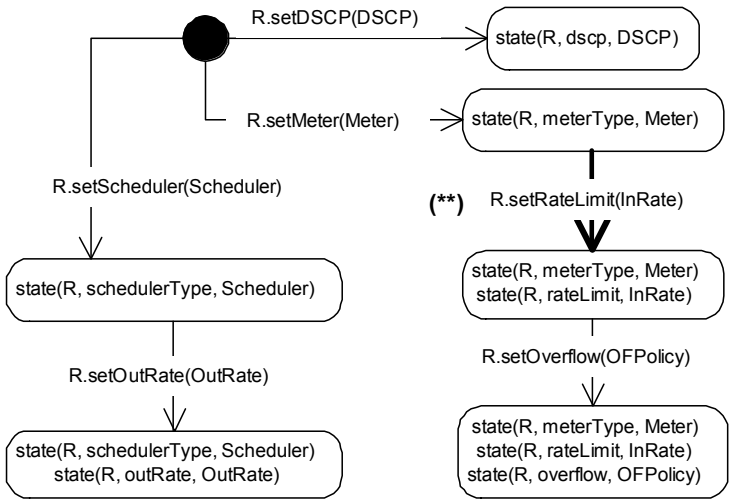


Figure 7-12. UML statechart for a diffserv router.

It is possible to transform this state chart into the Event Calculus notation presented previously where the input shown on each transition arrow is the action being performed. For transition between different states, the current state values become the `PostFalse` fluents; any actions associated with the transition and next state values become the `PostTrue` fluents; and the current state values become the `PreConditions`. Self-transitions should not specify the current state as `PostFalse` fluents. So following this scheme, the transition labelled (**) in Figure 7-12 would be represented in the Event Calculus as:

```
initiates(
 doAction(_, op(diffServRouter,
                        setRateLimit(inRate))),
 state(diffServRouter,rateLimit, inRate), T) ←
 holdsAt(pos(state(diffServRouter,
                   meterType, Meter)), T).
```

7.3.2 An Example

Network Quality of Service management requires administrators to manage the network devices and infrastructure to achieve predictable performance. The Differentiated Services (DiffServ) architecture can achieve this by aggregating network traffic into defined classes of service, and configuring routers to treat each of these classes appropriately. In such a network a packet might be handled differently at each hop based on the DiffServ class to which it belongs. Policy based management provides the ability to dynamically configure a system, by separating the rules that govern a system's behaviour from the functionality supported by it. Policies can be specified, and applied to large numbers of devices uniformly. In DiffServ, policies can be used to dynamically reconfigure routers so that the desired QoS goals are achieved as well as to perform admission control. It is important to be able to analyse policies to ensure consistency and to ensure that key properties are preserved in the network configuration, e.g. traffic marked in the same way is not allocated to different queues. Although adaptation can be realised through general scripting languages, policy languages adopt a more succinct, declarative, form in order to facilitate analysis. Many policy languages have

been proposed, but techniques for refining high level goal into implementable policies, amenable to analysis for consistency, remain poorly explored. Unless such techniques are developed and used in network management and provisioning tools, the additional expense required to deploy policy based management will remain difficult to justify.

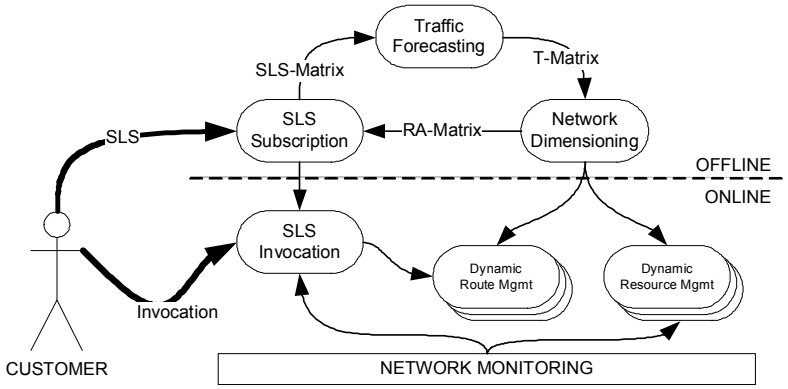


Figure 7-13. The TEQUILA diffserv QoS management framework operates in an offline mode to determine the network configuration required to meet long term traffic needs; and an online mode that adapts to short-term variations.

The Service Level Specifications (SLSes) which have to be satisfied by the network, as well as the derived QoS policies required to satisfy the SLSes, will change frequently. This process of deriving policies from the SLSes is recognized as one of the most difficult research challenges and is not fully automatable; however, techniques and tool support for refinement can be developed. Tool support to assist administrators in the refinement of policies would significantly reduce and improve network administration tasks especially when combined with analysis tools to ensure that only consistent specifications are derived. Although generic automated refinement is not achievable, useful, semi-automated tools can be achieved by constraining the problem to a well defined functional area, such as QoS management, where application specific knowledge can be encoded and used.

The TEQUILA framework operates in two modes – an offline mode that determines the configuration required to meet long-term traffic demands; and a run-time mode that adapts the configuration to meet short-term traffic variations. It can be decomposed into three sub-systems: SLS subscription, Traffic Engineering and Monitoring. SLS subscription is responsible for agreeing the customers' QoS requirements in terms of SLSes, while Traffic Engineering is responsible for fulfilling the contracted SLSes by deriving the network configuration. The Monitoring subsystem provides the above systems with the appropriate network measurements and assures that the contracted SLSes are indeed delivered at their specified QoS. Figure 7-13 shows a logical representation of this architecture. We focus here only on the behaviour of the Service Level Specification subscription (SLS-S) and Dynamic Resource Management (DRsM) components which are used in the scenarios presented in the next section.

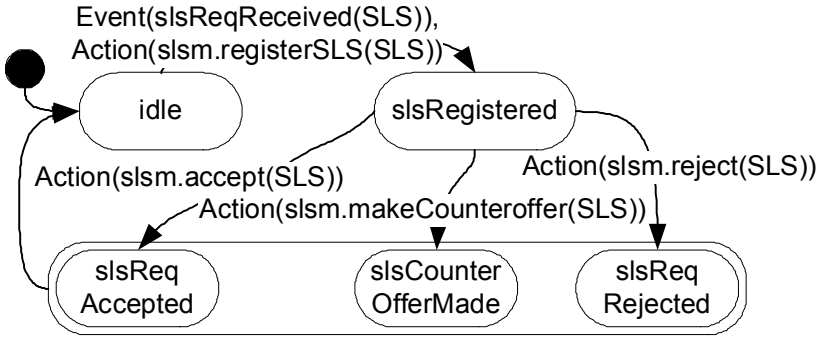


Figure 7-14.The Service level specification subscription (SLS-S) module is responsible for handling new SLA subscription requests, deciding to accept, reject or make counteroffers based on the policies of the QoS management system.

The SLS-S module performs admission control, calculates counter-offers and updates traffic forecasts using policies and so is the most relevant component for policy refinement. The SLS-S module uses the parameters of each requested SLS to calculate the expected traffic load based on traffic demand forecasts. This traffic is then aggregated with the expected traffic accumulated from the SLSes established during this Resource Provisioning Cycle (RPC). The resulting aggregated traffic defines the maximum potential demand and is mapped against the corresponding entries of the resource availability matrix (RA-Matrix). The result of this mapping is used by the admission control algorithm when deciding whether requests should be accepted or rejected. Requests are rejected if the risk of overwhelming the network with traffic so that QoS cannot be guaranteed is too high. A state chart model of this behaviour is shown in Figure 7-14.

The DRsM module is a distributed component responsible for reconfiguring the routers in response to short term variations in traffic. It is triggered by network monitors that track PHB utilization and raise threshold-crossing alarms when the bandwidth consumed by a PHB exceeds an upper threshold or drops below a lower threshold. Two values could be used for each threshold (trigger and clear values) to avoid repeated alarms when small oscillations occur. Once an alarm is raised, the DRsM calculates a new bandwidth allocation and configures the link appropriately; or triggers a new resource provisioning cycle (RPC) if sufficient bandwidth cannot be allocated. Policies determine how to calculate the new values, configure the link or trigger a new RPC.

Given the rules describing a system (SD) and the definition of some desired system state (i.e., the goal - G), abductive reasoning allows us to derive the facts that must be true for the desired system state to be achieved. As the goal is represented by a desired system state, the abductive reasoning process is essentially deriving a path in the statechart from some initial state to the desired one. This path is the derived strategy and can be represented using the following syntax:

Strategy	AchievedGoal
OnEvent	Events derived from transitions with system events.
DerivedActions	Actions derived from transitions with operations.
Constraints	Constraints derived from guards.

So following this scheme, the table below shows the formal representation derived from the state chart representation of the SLS-S module shown in Figure 7-14. Here Rule 1

defines the initial state of the SLSModule object to be 'init'. The next rule specifies that the occurrence of the reqReceived(sls) event and the registerSLS(sls) event cause the state to become 'slsRegistered'. This transition requires that the object ceases to be in the 'init' state and this is specified by the terminates(…) predicate in Rule 3. The remaining transitions of the statechart are defined in the similar manner in Rules 4-9. These rules also show how we make use of the pos() function described previously.

```
1-     initiallyTrue(pot_state(Obj, status, 'init')) ←
              mgdObj(Obj, 'classSLSModule').

2-     initiates(doAction(_, op(Obj, registerSLS, parms(sls))),
              pot_state('d_mgdObjsd_slsmgr', status, 'slsRegistered'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'init')), T),
                     happens(sysEvent(reqReceived(sls)), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

3-     terminates(doAction(_, op(Obj, registerSLS, parms(sls))),
              pot_state(Obj, status, 'init'), T) :-
                     holdsAt(pos(pot_state(Obj, status, 'init')), T),
                     happens(sysEvent(reqReceived(sls)), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

4-     initiates(doAction(_, op(Obj, makeCounteroffer, parms(sls))),
              pot_state(Obj, status, 'slsCounterofferMade'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

5-     terminates(doAction(_, op(Obj, makeCounteroffer, parms(sls))),
              pot_state(Obj, status, 'slsRegistered'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

6-     initiates(doAction(_, op(Obj, reject, parms(sls))),
              pot_state(Obj, status, 'slsReqRejected'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

7-     terminates(doAction(_, op(Obj, reject, parms(sls))),
              pot_state(Obj, status, 'slsRegistered'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

8-     initiates(doAction(_, op(Obj, accept, parms(sls))),
              pot_state(Obj, status, 'slsReqAccepted'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).

9-     terminates(doAction(_, op(Obj, accept, parms(sls))),
              pot_state(Obj, status, 'slsRegistered'), T) ←
                     holdsAt(pos(pot_state(Obj, status, 'slsRegistered')), T),
                     mgdObj(Obj, 'classSLSModule'),
                     time(T).
```

The technique for policy refinement presented here is based on formal methods, which by their very nature can be difficult to use. The formal specification of the system and policies can be particularly verbose and the results generated from the strategy derivation process are not easy to interpret. Therefore it is important that adequate tool

support is provided for administrators and that the tool developed helps them to easily specify the organization and behaviour of the managed system, together with the goals and policies that apply.

A key requirement of the tool is to make the underlying formal notation and reasoning techniques usable. To this end, we allow all information regarding managed objects and their behaviours to be specified in UML using any available editor (e.g. Rational Rose, ArgoUML etc). Then, by using the standard XML Meta-data Interchange (XMI) format of UML, it is possible import these specifications into the policy analysis and refinement tool. This tool can handle the specification of policies, goals, domain hierarchy and the generation of analysis results.

Figure 7-15 shows the overall architecture of the tool for policy analysis and refinement. As can be seen, there are 3 principal components – the domain service; the analysis service; and the user-interface client application.

Domain Service The domain service provides functionality for storing and retrieving information that describes the entire managed system. In addition to the domain hierarchy itself, this includes the policies, managed objects and goals. Figure 7-15 shows a detailed class diagram of the objects stored in the domain service, together with the programming interface provided to the other components for accessing the domain hierarchy. The base class for all types of information stored in the domain service is called ManagedEntity and this class has entityID, name, domainPath and description attributes which are inherited by all the subclasses. For a given managed entity instance, the entityID attribute is assumed to be a unique identifier.

Additionally, results from analysis activities are stored in objects of type AnalysisResult and associated with the managed entity to which they apply. As shown, a managed entity can be a domain, policy, managed object or goal.

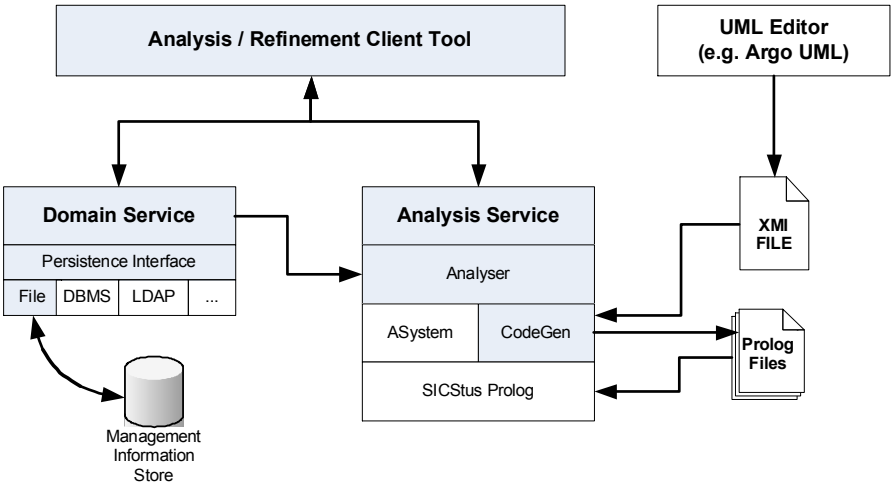


Figure 7-15. Architecture of the policy refinement and analysis tool. The tool consists of 3 components – the domain service, the analysis service and the user-interface client tool.

Analysis Service The analysis service deals with the requirements of translating the high-level representations of the policy based management system into the underlying formalism and generating the analysis and refinement results. To do this the analysis service is integrated with the SICStus Prolog system which provides deductive reasoning capabilities. Abductive reasoning is provided by the A-System abductive proof engine [69], which runs within the SICStus Prolog environment.

Translation of the high-level representations of the domain hierarchy, managed object behaviours, policies and goals is achieved through the use of XML style sheet transformations (XSLT). The AnalysisServer class is implemented as a Java RMI server providing methods for analysis tasks like retrieving policy conflict data and performing review queries. Whenever a change is made to information stored in the domain service, the analysis server uses the Analyser class to initiate the process of generating the required Prolog code. The actual translation functions are implemented in the CodeGen component (Figure 7-15). The AnalysisServer is also used to generate the decompositions and strategies for high-level goals.

Analysis and Refinement Client The final component of the analysis and refinement tool is the client application that implements the user interface (Figure 7-16). The primary view of this application is based on a new version of the Ponder hyperbolic domain browser [70]. It uses the HyperGraph library to implement a hyperbolic viewer to browse a hierarchical domain structure more effectively than a simple tree viewer. It also supports the option of selectively collapsing certain sub-domain hierarchies.

In order to view the detailed information regarding any particular entity in the domain hierarchy, a context sensitive properties pane was used. Whenever the user selects an element of the domain hierarchy, the tabbed panel in the bottom half of the screen shows the details relating to that particular element. For example, Figure 7-16 illustrates the goal decomposition detail panel, showing the sub-goal information, together with the derived strategies. At present, the tool is a research prototype that aims to demonstrate the practical applications of our approach to policy analysis and refinement. For this reason, some of the specifications shown in the user interface are still presented in the formal notation. However, we expect usage to be considerably simplified in production versions of the tool.

In the next section we present two scenarios where the policy refinement approach and tool described are used to derive a policy to meet the requirements of the TEQUILA framework.

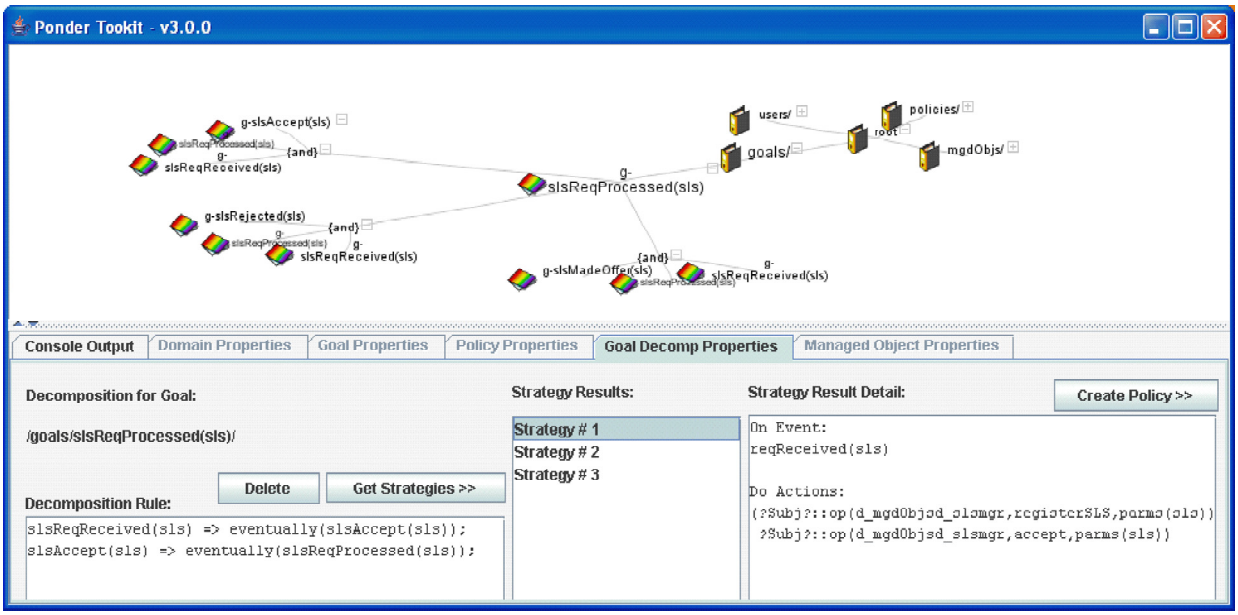


Figure 7-16. Policy refinement tool user interface. The top pane displays the domain hierarchy and the bottom pane has a number of detail tabs. Here we show the detail of a goal decomposition, together with the strategies derived. The decomposition rule (bottom left) describes the relationship between the sub-goals.

7.3.3 Example 1: Admission Control

Consider an example where a new SLS from the customer, AOL, requires a pipe between routers R1 and R6 with Expedited Forwarding (EF) per hop behaviour, 20ms delay, zero packet loss, and a 10Mbps throughput guarantee. SLS[customer: aol; scope: pipe(r1,r6); qos: qosClass(EF, 20, 0); bwReq: bw(10Mbps)], is presented to the SLS-S subscription module. The SLS-S module registers the SLS, compares its contents with the Resource Availability (RA)-Matrix and decides whether to accept, reject or make a counteroffer. Policies are used to influence the choice of the SLS-S module. The policy that applies depends on the goals that need to be achieved. For example, the highest level goal below ensures that the SLS request is processed:

*G*1: *Goal* SLSRequestProcessed
FormalDef slsReqReceived(SLS) ⇒ slsRequestProcessed(SLS).

Since applying the abductive analysis to the system description of the SLS-S module does not produce strategies for achieving this goal, it is necessary to elaborate it further by the domain-independent pattern GP2' (see below) to decompose the above goal into the following sub-goals. In each case we use abduction to derive a strategy:

*G*2: *Goal* SLSRequestAccepted
FormalDef slsReqReceived(SLS) ⇒
slsReqAccepted(SLS) ∧(slsReqAccepted(SLS) ⇒ ◇slsRequestProcessed(SLS)).
*G*3: *Goal* SLSRequestRejected
FormalDef slsReqReceived(SLS) ⇒
slsReqRejected(SLS) ∧(slsReqRejected(SLS) ⇒ ◇slsRequestProcessed(SLS)).
*G*4: *Goal* SLSCounterofferMade
FormalDef slsReqReceived(SLS) ⇒
slsCounterofferMade(SLS) ∧ (slsCounterofferMade(SLS) ⇒ ◇slsRequestProcessed(SLS)).
*S*1: *Strategy* G2: SLSRequestAccepted
OnEvent slsReqReceived(SLS)
DerivedActions slsm.registerSLS(SLS) -> slsm.accept(SLS).
*S*2: *Strategy* G3: SLSRequestRejected
OnEvent slsReqReceived(SLS)
DerivedActions slsm.registerSLS(SLS) -> slsm.reject(SLS).
*S*3: *Strategy* G4: SLSCounterofferMade
OnEvent slsReqReceived(SLS)
DerivedActions slsm.registerSLS(SLS) -> slsm.makeCounteroffer(SLS)

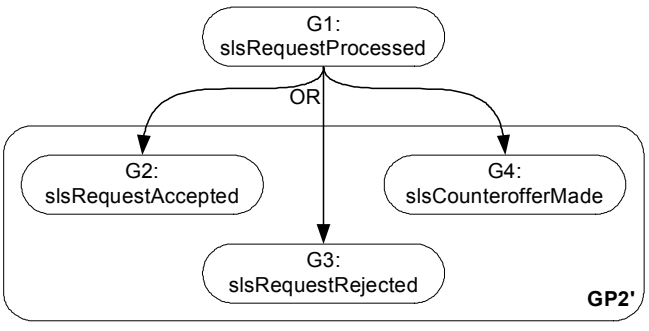


Figure 7-17. Goal decomposition for SLS subscription scenario

As shown in Figure 7-17, goal elaboration yields a disjunction of goals (G2-G4), and the user can select the sub-goal that best satisfies the requirement. Strategies (S1-S3) are derived automatically and identify the action sequences (->, sequence operator) that achieve each of the sub-goals. In this scenario, the required high-level policy is that SLS requests from customer 'AOL' with qosClass(EF, 20, 0) should be accepted if the bandwidth requested is less than the bandwidth available in the RA-Matrix for the same QoS class. As this policy achieves the SLSRequestAccepted goal we can encode the corresponding strategy into a policy as follows:

```
P1: inst oblig /policies/slsm/acceptAOLSLS_P1 {
          on    slsReqReceived(SLS);
          subj  s = /slsmPMA;
          targ  t = s.slsm;
          do    t.register(SLS) -> t.accept(SLS);
          when  SLS.customer = 'aol' &&
                 SLS.qosClass = qosClass(ef, 20, 0) &&
                 t.getAvailBW(SLS.qosClass) > SLS.bwReq; }
```

Whilst the strategy is derived automatically, user intervention is required to map the event and constraints specified in the goal into the policy. Additionally, the system helps the user select the specific subjects and targets by automatically identifying objects of the required types in the domain hierarchy. Thus, the high-level goal specified by the network administrator is refined into a concrete policy.

Figure 7-16 shows how this scenario would appear in the policy refinement toolkit. Here the top panel illustrates the goal elaboration hierarchy within the organizational domains of the system. The bottom panel shows the detail of one of the goal decompositions, where the relationship between the sub-goals is shown by the decomposition rule in the bottom left, and the strategies derived are shown to the right.

7.3.4 Example 2: Adapting to Traffic Increase

This scenario illustrates how the TEQUILA framework responds to short-term traffic changes from customers. The network administrator wants to ensure that when such an increase occurs between 11am and 1pm and causes a network utilisation greater than 85% of the maximum allocation, the bandwidth allocation should be increased by 10% and spare capacity should be equally split amongst the Per-Hop-Behaviours (PHBs). In this situation the Dynamic Resource Management (DRsM) module at each link along the traffic route would respond as follows:

1. On receiving a traffic increase alarm, the DRsM decides on the appropriate action to adapt to the increase using guideline values for maximum, minimum and congestion bandwidth allocations provided by the ND.
2. Configure the link/PHB with this new value and decide on how to allocate any spare link capacity amongst all the link/PHBs.

Policies are used at each of the stages above, to decide how to calculate the new bandwidth allocation, and how to distribute spare link capacity. In each case the exact policy to be used depends on the required goal. For the policy decisions on calculating the new bandwidth allocation and then allocating spare capacity, the high-level goal (G6) should achieve the state "adapted configuration" when an alarm is raised. This can be stated as follows:

*G*6: *Goal* ConfigAdaptedForBWUtilIncrease

FormalDef alarmRaised(bwUtilIncr, [utilValue, PHB]) ⇒ ◇ configAdapted.

In this case the abductive analysis of G6 yields no strategy, so the goal must be elaborated further. Applying GP2' yields the sub-goals NewRPCRequested (G7) or CalculatedConfigNewBWAllocation (G8). Each of these goals leads to the high-level goal G6 being satisfied as shown in their formal definitions below.

*G*7: *Goal* NewRPCRequested

FormalDef alarmRaised(bwUtilIncr, [utilValue, PHB]) ⇒

requestedNewRPC ∧(requestedNewRPC ⇒ ◇ configAdapted).

*G*8: *Goal* CalculatedConfigNewBWAllocation

FormalDef alarmRaised(bwUtilIncr, [utilValue, PHB]) ⇒

calcAndConfigNewBWAlloc ∧ (calcAndConfigNewBWAlloc ⇒ ◇ configAdapted).

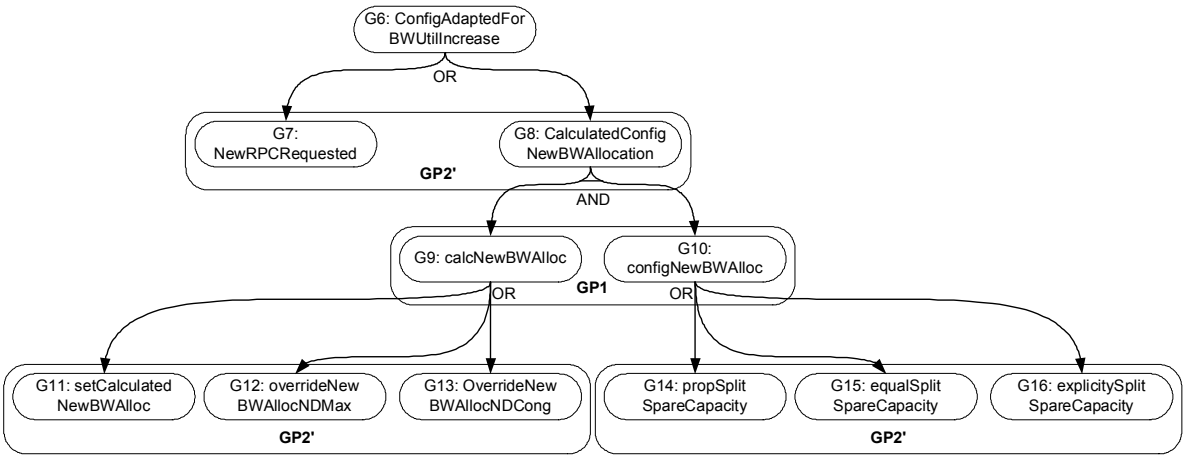


Figure 7-18. Goal decomposition for traffic increase scenario.

In the scenario, the high-level policy requires calculating and configuring a new bandwidth allocation, represented by goal G8 above. However, since it is not possible to automatically derive a strategy for this goal, it is necessary to elaborate it further, this time using a combination of the patterns GP2' and GP1. Figure 7-18 indicates the applicable patterns at each stage with the following goals:

*G*9: *Goal* calcNewBWAlloc
FormalDef calcNewBWAlloc(newValue) ⇒ ◇ configNewBWAlloc.
*G*10: *Goal* configNewBWAlloc
FormalDef configNewBWAlloc ⇒ ◇ configAdapted.
*G*11: *Goal* setCalculatedNewBWAlloc
FormalDef calcNewBWAlloc (newValue) ⇒ (newValue = calcValue)
∧ (newValue = calcValue) ⇒ ◇ configNewBWAlloc.
*G*12: *Goal* overrideNewBWAllocNDMax
FormalDef calcNewBWAlloc (newValue) ⇒ (newValue = drsm.ndMaxBWAlloc)
∧ (newValue = drsm.ndMaxBWAlloc) ⇒ ◇ configNewBWAlloc.
*G*13: *Goal* overrideNewBWAllocNDCong
FormalDef calcNewBWAlloc (newValue) ⇒ (newValue = drsm. ndCongBWAlloc)
∧ (newValue = drsm.ndCongBWAlloc) ⇒ ◇ configNewBWAlloc.
*G*14: *Goal* propSplitSpareCapacity
FormalDef configNewBWAlloc ⇒ spareCapProportionallySplit
∧ spareCapProportionallySplit ⇒ ◇ configAdapted.
*G*15: *Goal* equalSplitSpareCapacity
FormalDef configNewBWAlloc ⇒ spareCapEquallySplit
∧spareCapEquallySplit ⇒ ◇ configAdapted.
*G*16: *Goal* explicitySplitSpareCapacity
FormalDef configNewBWAlloc ⇒
spareCapExplicitlySplit([splitValues]) ∧
spareCapExplicitlySplit([splitValues]) ⇒ ◇ configAdapted.

In this scenario, the goals of the administrator are G11 and G15. So, we are interested in the strategies for setting the new bandwidth to the newly calculated value and splitting spare capacity equally. Performing the abductive analysis on the statechart representation of the DRsM calculation and configuration module behaviours yields the following strategy, which in turn can be encoded into a policy:

```
S5: Strategy G11: setCalculatedNewBWAlloc &&
              G15: equalSplitSpareCapacity
     OnEvent          alarmRaised(bwUtilIncr, [utilValue, PHB])
     DerivedActions   calcValue = drsm.incrAllocBW(PHB, pct) ->
                      drsm.configureLink(PHB, calcValue) ->
                      drsm.splitSpareCapEqually
     Constraints      drsm.incrAllocBW(PHB, pct) <
                      drsm.ndMaxBWAlloc(PHB).

P3: inst oblig /policies/adaptTrafficIncreaseAOLSLA_P1 {
         on    alarmRaised(bwUtilIncr, [utilValue, ef]);
```

```
        subj   s = /routers/FromR1/ToR6/drsmPMAs/;
        targ   t = s.drsm;
        do     calcValue = t.incrAllocBW(ef, 10) ->
               t.configureLink(ef, calcValue) ->
               t.splitSpareCapEqually;
        when   t.incrAllocBW(ef, 10) < t.ndMaxBWAlloc(ef) &&
               time.between('11:00', '13:00');
    }
```

Note that the abductive analysis results in a strategy that includes constraints. These are derived from the guards defined in the state chart of the system behaviour and must therefore be included in addition to any other constraints manually mapped from the high-level policy. This is illustrated in policy P3, which combines the strategy constraint with the time constraint from the high-level policy.

7.3.5 Summary

We have shown how abductive reasoning can be used for deriving the Ponder policies that can achieve the refined goals. Tool support is provided in the form of an integrated tool set that hides the underlying complexity of the formal representation and allows Adminstrator Designers and Administrator Consultants to manipulate goals, properties and design level models of the managed system. The approach based on abductive reasoning also provides the ability to encode the policies derived as goal refinement patterns that can be added to the database and reused in subsequent iterations. This approach permits to reason with partial knowledge and in particular partial system descriptions. This is particularly useful in networks and systems management where higher level management functionality can be provided through network management platforms such as CA Unicenter, HP Openview or Tivoli.

7.4 A Model-Checking Based Approach to Enforcement of System Behaviour

We present in this section a second approach to the derivation of policies from the refined goals which is based on model checking and the recent advances in the software engineering community on the development of model-checking tools such as SPIN, PVS and others that have been shown to scale to larger system descriptions.

This process is aimed at imposing a given behaviour on the managed entities to accomplish the specifications provided by the “Establishment of Temporal Relationships” process (section 7.2.4.1.1 - Figure 7-7).

This sub-process relies on automated mechanisms grounded in AI planning techniques, particularly through Model Checking [71], [72]. More concretely, as our “Establishment of Temporal Relationships” technique is by means of LTL, we use the concept of Planning by Model Checking with LTL in the context of our refinement framework [73].

In general terms, plans are understood as sequences of actions that lead one system from an initial state to a target state [71]. In the Planning by Model Checking with LTL technique, our time related goals are expressed by means of LTL formula and LTL

Model Checking is used to determine plans that indicate how such system should behave for satisfying the goals [74], [66], [75].

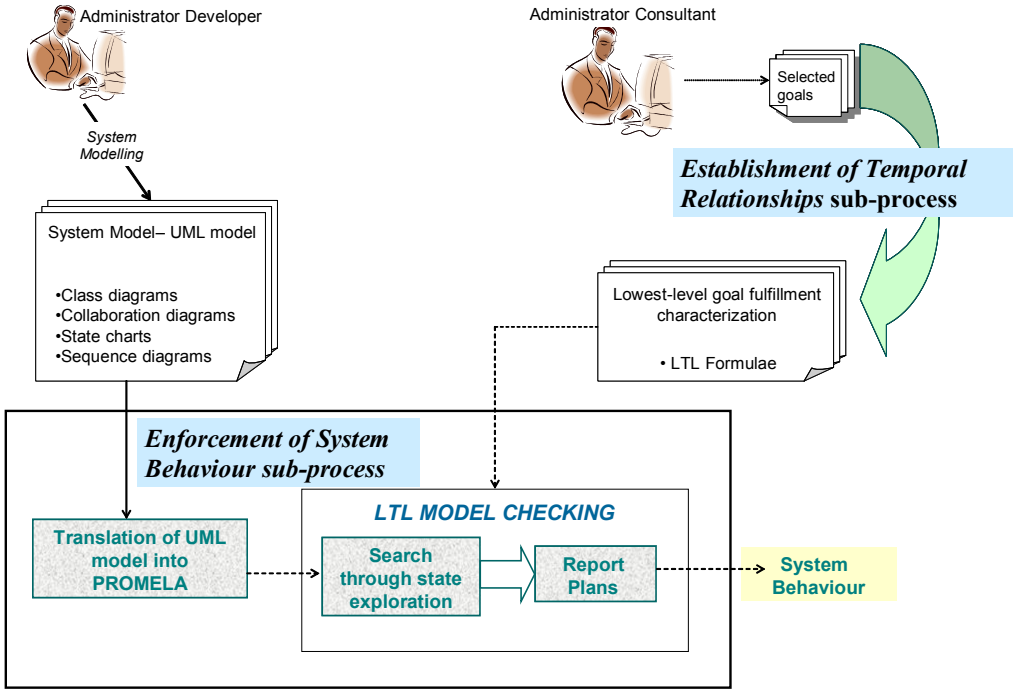


Figure 7-19. The planning by model checking approach.

As a Model Checking-based approach, the Enforcement of System Behaviour process has two inputs; namely the System Model and the established temporal relationships amongst lowest-level goals, i.e. the output of the process *Establishment of Temporal Relationships*. Regarding the system model, we consider that the Administrator Developer uses for this purpose standard UML notations such as class diagrams, collaboration diagrams, state charts and sequence diagrams. Given that the main process relies on LTL-based state exploration via Model Checking, this process includes a translation mechanism of the UML models into PROMELA, the language interpreted by the SPIN Model Checking engine, as shown in Figure 7-19.

The SPIN searching engine explores the state space finding the necessary transitions the system must take to reach the sequence of states as prescribed by the LTL formulae. The SPIN searching engine then provides a plan of transitions to be executed by the model entities. The process ends verifying that the plan includes the states prescribed by the lowest-level goals and that these states are reported in the ordering prescribed by the LTL formulae. Figure 7-20 shows a graphical representation of a plan obtained in the *Enforcement of System Behaviour* sub-process which in turn corresponds to a SPIN system trace report.

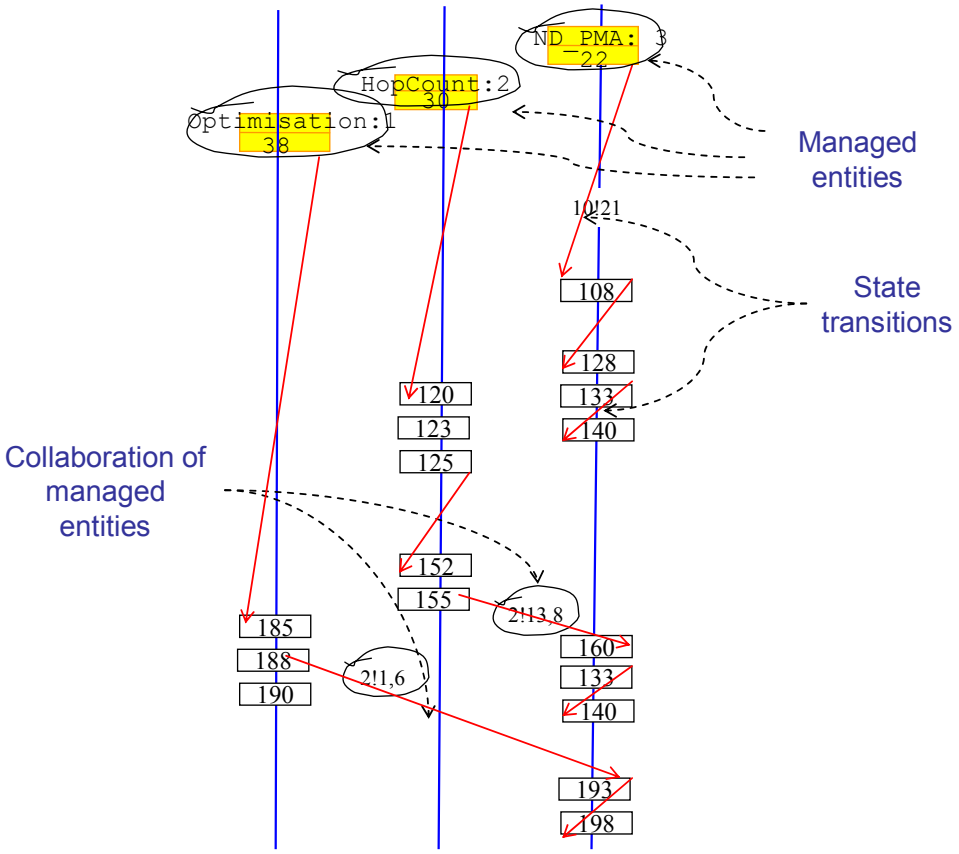


Figure 7-20. Graphic output sample of the enforcement of system behaviour.

The system trace executions provided by SPIN identify the managed entities in charge of executing the plan of actions (see *Managed entities* in Figure 7-20). In addition, the traces allow identifying the state transitions, at every point of the execution, which the corresponding Managed entities should experiment during the plan (see *State transitions* in Figure 7-20). Worthy of note is that plans satisfying the established temporal relationships of lowest-level goals may require the collaboration of several model entities. The plans reported by the SPIN searching engine allow identifying how the model entities collaborate during state transitions. This situation occurs for example when an entity transitions from one state to another as a result of the reception of a notification from another entity as illustrated by the pointer *Collaboration of managed entities* in Figure 7-20.

Since the plans include every transition at every point of the execution, the Model Checking reports do not identify the state transitions that should be controlled by policies from those inherent of the system operation. Therefore, further analysis is necessary to acquire meaningful policy information from the reported trace execution. The following section introduces the concept of Translation Process [76] that we have developed on purpose to abstract policy information from system trace executions.

7.4.1 The Translation Process

This process is intended to identify the policies that would be necessary to reproduce the system trace execution reported by the *Enforcement of System Behaviour* process. This process relies on the principle that policies control system state transitions [77] and that the reported plans above mentioned include, among others, the policy-controlled state transitions. A Translation Process [76] is used to abstract the policy-related

information from policy-controlled transitions. In this sense, it is mandatory that the developer provides the state transitions that are controlled by policies during the design/development of the system.

The first step towards the application of Translation Process is the identification of *transition plans*. A *transition plan* (TP) is a sub-section of a system trace execution that includes a policy-controlled state transition and that is characterised as follows:

- PSi a pre-condition in a managed entity S
- PQi a pre-condition in a managed entity Q
- $T_{Si,Si+1}$ a state transition in the managed entity *S*
- $T_{Qi,Qi+1}$ a state transition in the managed entity *Q* that we want to make policy-controlled as a result of transition $T_{Si,Si+1}$

Then, the transition plan TP is represented as TP=[PSi,TSi,Si+1 => PQi,TQi,Qi+1] and its semantics is: "on the occurrence of PSi in the managed object S, preceding the transition $T_{Si,Si+1}$, the managed object Q must enforce the transition $T_{Qi,Qi+1}$". A graphical representation of a transition plan TP is illustrated in the left part of Figure 7-21.

The Translation Process materialises the above prescription with Event-Condition-Action (ECA) policies. For this, we have used the Ponder obligation policy structure [78]. Obligation policies are event-triggered Condition-Action rules that define the activities subjects must perform on objects in a target domain.

In the context of our refinement approach we consider that a policy-controlled transition is the result of an action enforcement within the target domain. Hence, transitions are interpreted as actions executed on a managed object. A graphical representation of the Translation Process is shown in the right part of Figure 7-21**.** This mapping is done following the Ponder prescription that Obligation policies are event triggered and define the actions that subjects must perform on objects of the target domain.

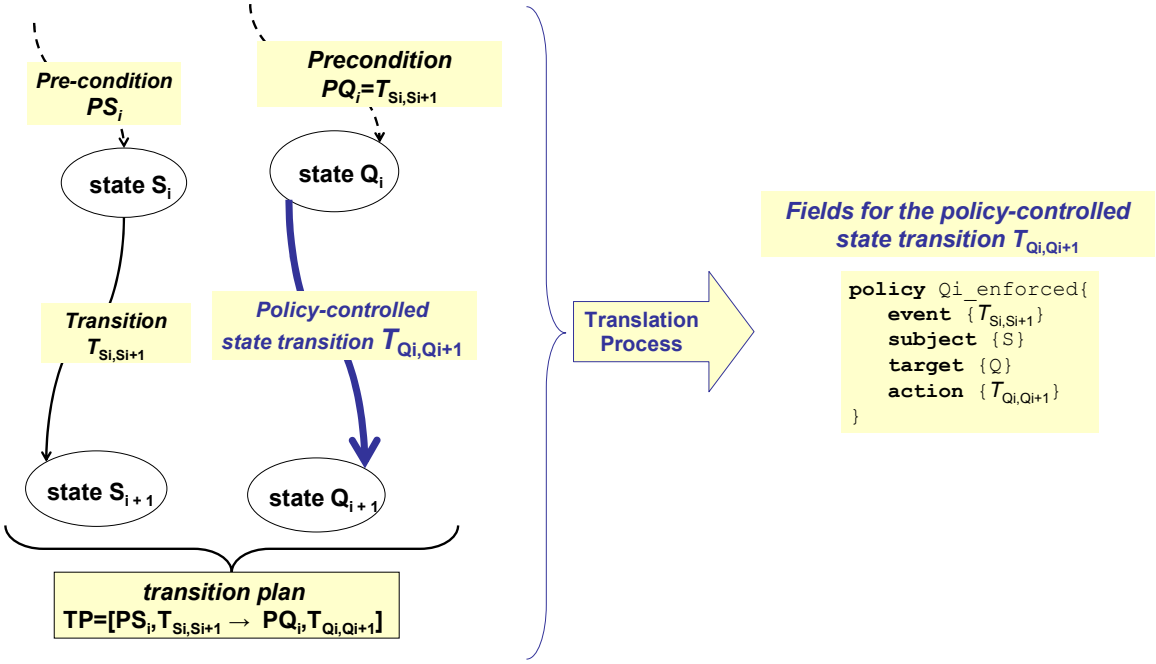


Figure 7-21. Representation of the translation process.

Typical outputs of the *Enforcement of System Behaviour* process may include more than one policy enforceable transition (e.g. transition $T_{Qi,Qi+1}$) and consequently multiple transition plans. The transition plans are mapped to their corresponding policy fields by applying the Translation Process. Consequently, the result of the *Translation Process* sub-process is a set of policy fields (e.g. policy $Q_{i\_enforced}$ in Figure 7-21), one for each policy-controlled transition.

7.4.2 Encoding of Deployable Policies

In order to deploy the abstracted policies onto the managed system, subjects and targets identified above must be matched to the actual object distribution of the system. This is the aim of this final refinement process.

The policy refinement framework considers that the Administrator Developer should document the Object Distribution (object model) during system design and development. The *Encoding of Deployable Policies* process makes use of this information to encode deployable policies from the policy fields provided by the *Translation Process*. Figure 7-22 shows a graphical representation of the final step within the refinement process scenario. This process is achieved in a fully automated manner once the object distribution documentation is available during the operation of the system.

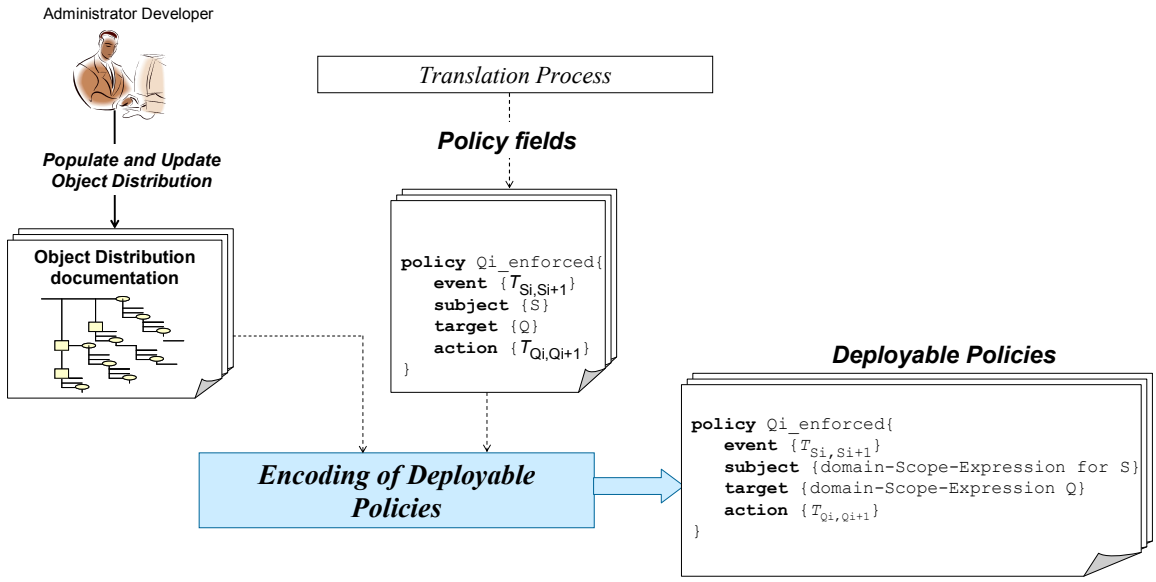


Figure 7-22. Encoding deployable policy sub-processes.

7.4.3 An Example for Autonomic Network QoS Management

Policy based management has been regarded as a suitable approach to manage and control complex systems and is widely considered as an enabling technology of Autonomic Management. Despite its potential benefits of flexibility and constrained programmability, this solution is still not a reality. In addition to the crucial problem of policy conflicts, a key issue behind the reluctance to adopt this technology is the necessity to derive executable policies aligned with administrative guidelines. The latter is normally referred as the *policy refinement problem*. The methodological approach described in the previous sections can provide a solution to this problem and offer another tool towards the realisation of Autonomic Management.

7.4.3.1 Applying Autonomic Management Principles

In order to utilise the methodology effectively we need to consider its relations with the principles of Autonomic Management. Policy based management can effectively address the issues of Self-configuration and Self-optimization as two of the key properties of Autonomic Systems. These are illustrated in a case study that will be presented in the subsequent section, demonstrating a realistic application of the methodology for the provisioning of Quality of Service in IP Networks.

Self-configuration is a key property of Autonomic Systems and is widely accepted that policy based management can address its requirements effectively. Towards that direction, a policy refinement methodology adds an extra layer of automation by simplifying the process of creating enforceable low level policies. In addition, the system is able to select the optimal method in terms of selected policies, in order to achieve the defined high level goals. This procedure is in essence Self-Optimisation of the system, as the best available combination of policies is enforced without additional human intervention.

7.4.3.2 A Quality of Service Management Approach

Discussions about Autonomic Management often remain theoretic and there is an obvious lack of solutions and proofs of concept. In this section, we present a realistic case study implementation that on one hand validates the presented methodological approach for policy refinement and on the other illustrates how the concepts of Autonomic Management can be realised.

We initially describe the application domain that we have used for this purpose and then define a policy hierarchy and realistic high-level goals for this domain. Last, we show the practicality of our approach using our framework implementation.

The QoS Management solution in which we have validated our approach relies on the principles developed in the context of the TEQUILA project. A simplified representation of this approach is depicted in Figure 7-23a., which shows the integration of Service Management and Traffic Engineering functions to achieve Quality of Service provisioning in IP Networks.

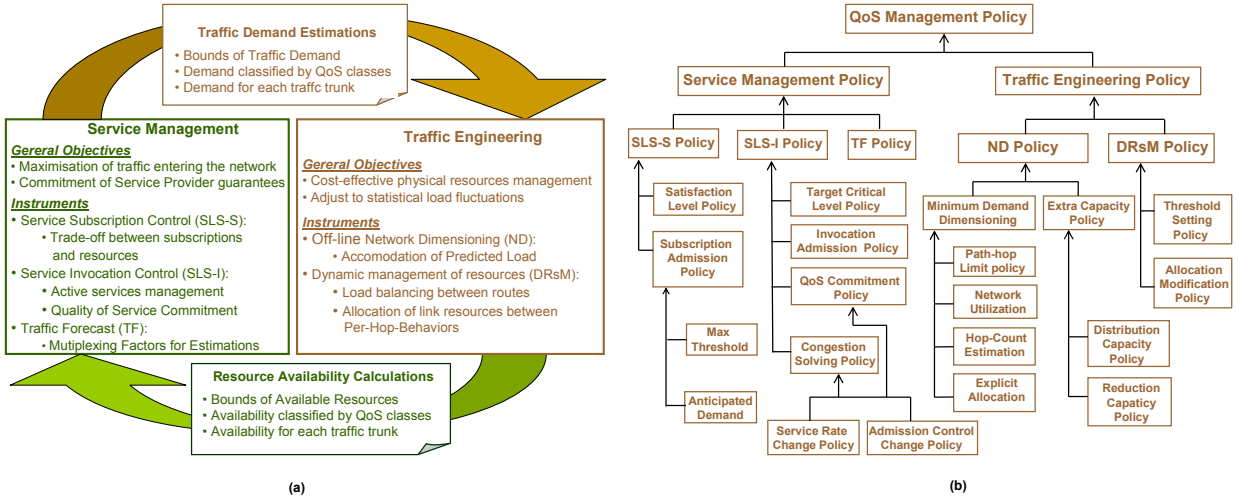


Figure 7-23. (a) Generic approach for QoS management; (b) Policy hierarchy for QoS management.

The Service Management functionality has two objectives: the maximization of traffic entering the network, and the commitment of the service provider's QoS guarantees. As

the traffic entering the network is a function of the number of subscribed contracts and active services, admission control mechanisms are defined for service subscriptions and invocation requests. QoS commitment is addressed by enforcing preventive and corrective actions as a means to police misbehaving users and to resolve congestion.

The Traffic Engineering functionality is concerned with the management of resources. An off-line dimensioning process is responsible for mapping the predicted traffic demand to physical network resources. In addition, real-time operations are implemented as a means to first, balance the load amongst the established Label Switched Paths (LSPs) in the network, and second, to ensure that link capacities are appropriately distributed among the different Per-Hop-Behaviours (PHBs) sharing each link.

7.4.3.2.1 A Policy Hierarchy for QoS Management

Following the principles of our methodological approach, a QoS Management Policy must consider both, Service Management and Traffic Engineering policies given that QoS delivery involves these two functions. A policy hierarchy for QoS Management is shown in Figure 7-23b for which we provide a brief description.

The Service Management functions are carried out by three components: Service Subscription (SLS-S), Service Invocation (SLS-I) and Traffic Forecast (TF), defining respectively the *SLS-S Policy, SLS-I Policy* and *TF Policy*.

The *SLS-S Policy* offers the necessary programmability to control service subscriptions; acceptance, negotiation, rejection. It controls the trade-off between the number of subscriptions and the confidence for ensuring a given QoS. It is enforced by two lower-level policies: the Satisfaction Level Policy and the Subscription Admission Policy.

- The Satisfaction Level Policy is used to define levels of confidence for service fulfilment.
- The Subscription Admission Policy deals with subscription admission control. Given that the admission control logic is a function of the maximum anticipated demand, this policy is enforced by two lowest-level policies: the Max Threshold, and the Anticipated Demand. These two have been derived as the means to define thresholds for admission control and the multiplexing factors that would influence the considerations for the total anticipated demand, and consequently the admission control logic.

The *SLS-I Policy* controls the number and type of active services and consequently, the volume of injected traffic. It addresses the trade-off between the number of admitted invocations and preventing QoS degradation due to network overloading. This policy controls different sub-functions defining four lower-level policies; Target Critical Level Policy, Invocation Admission Policy, QoS Commitment Policy, and Congestion Solving Policy.

- The Target Critical Level Policy defines the level at which the likelihood of overwhelming the network is considered critical.
- The Invocation Admission Policy controls the invocation admission functionalities.
- The QoS Commitment Policy prevents QoS degradation by enforcing proactive actions.

- The Congestion Solving Policy executes penalty actions to avoid congestion.

The enforcement of the latter two policies may result in service rate re-allocations and/or admission control re-adjustments for new invocations. These two define respectively the Service Rate Change Policy and the Admission Control Change Policy.

The *TF Policy* is used to define the criteria by which a service is considered to enjoy an "*almost satisfied"* and a "*fully satisfied"* rate, both used to define bounds of traffic demand estimations.

The Traffic Engineering functions are supported by the Network Dimensioning (ND) and Dynamic Resource Management (DRsM) components which in turn define the *ND Policy* and *DRsM Policy* in the hierarchy.

The *ND Policy* controls the accommodation of traffic estimations into the physical resources and is enforced by two lower-level policies; the Minimum Demand Dimensioning policy controls the strategy of allocation of the minimum estimated traffic demand and the Extra Capacity Policy controls the allocation of the remaining resources.

The *DRsM Policy* drives the dynamic resource management functions. It controls the criteria on how resources allocation should be re-defined when considerable load fluctuations take place. The lowest-level policies for the ND Policy and the DRsM Policy are shown in Figure 7-23b.

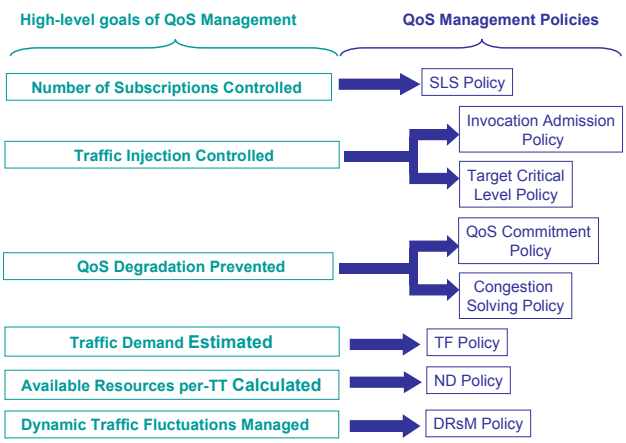
7.4.3.2.2 High-level Goals for QoS Management

Different methods and mechanisms can be found in the literature [79] to specify business objectives and to provide indicators to assess IT performance related to them. In the context of our scenario, these represent the high-level goals with which the administrator developer controls QoS provisioning.

In our QoS management scenario the administrator developer considers the following high-level goals: *Number of Subscriptions Controlled*, *Traffic Injection Controlled*, *QoS Degradation Prevented*, *Traffic Demand Estimated*, *Available Resources per-Traffic Trunk Calculated*, *Dynamic Traffic Fluctuations Managed*. The administrator developer should establish a relationship between these high-level goals with the achievement of the policies specified in the policy hierarchy.

For example, the high-level goal *Number of Subscriptions Controlled* is directly influenced by the Service Subscription Policy given that the latter is used to control the acceptance, rejection/negotiation of service subscriptions. This way, the developer establishes a relationship between the Number of Subscriptions Controlled high-level goal and the SLS Policy. Similarly, the high-level goal *Traffic Injection Controlled* is influenced by the traffic entering the network and the quality of service enjoyed by the active services. The latter two aspects are influenced by the Invocation Admission Policy and Target Critical Level Policy and consequently these policies are related to the Traffic Injection Controlled high-level goal. Figure 7-24a summarizes the high-level goals and their association(s) with the policy hierarchy in our scenario.

The interpretation of the above high-level goals would define a "particular" view for QoS Management. For example, the instantiations shown in Figure 7-24b provide an administrator consultant's view of QoS Management at system operation time. Another consultant could define different views according to previous experiences, statistical data, etc.



Number of Subscriptions Controlled	Available Resources per-TT Calculated	QoS Degratation Prevented	Traffic Demand Estimated	Dynamic Traffic Fluctuations Managed
Maximize subscriptions with the **Highest Confidence** to provide the agreed QoS	Allocate Minimum demand with **Average hop-count** estimations **Minimize** link overloading	**Mild Proactive** actions when traffic is **Half-way** between the resources **available at congestion** and the **maximum**	**Almost** and **Fully Satisfied** rates are **100%** the subscribed ones for **real-time traffic**	**Increase or decrease** resources by **10%** the range provided by ND **when load fluctuates 10%** the range provided by ND
Traffic Injection Controlled **Maximize the Traffic** injected into the network	Redistribute over-provisioning and spare capacity **proportionally**	**Prime Proactive** actions when the traffic injected is **Equal** to **maximum** available, and at **congestion**	Almost and Fully satisfied rates are **60% and 85%** respectivelly for **other type of traffic**	**Redistribute the changes proportionally** to every PHB Redistribute **spare capacity proportionally** to every PHB

(b)

Figure 7-24. (a) High-level goals and policy relationships for QoS management; (b) "particular" view of QoS provisioning management.

The first column, for example, concerns the view of QoS delivery with respect to the high-level goals *Number of Subscriptions Controlled* and *Traffic Injection Controlled*. For the former the consultant opts to maximize the number of subscriptions at the Highest Confidence levels. This directive implies that congestion occurrence would be highly unlikely and then the consultant has opted to maximize the traffic injected into to the network for the latter directive.

7.4.3.2.3 Policy Refinement Process for Autonomic Management

In order to show the practicality of the refinement process we have used a prototype of our framework which supports the following features:

- *Industrial support for Goal Management*. The prototype [76] integrates Objectiver, a formal tool for Goal-oriented specifications that provides support for Goal Refinement and Goal Selection.
- *Tool support for system documentation*. The prototype integrates an ArgoUML toolkit to document the managed system features in UML standard notations.
- *Support to acquire executable policies automatically*. Different tools and ad-hoc software modules specialise the automated Policy Refinement Mechanisms of the framework. The prototype includes:
- A Hugo/RT toolkit to allow translating system documentation into code for formal analysis.
- A SPIN searching engine to find the necessary system behaviour that fulfils specific system goals.
- A modified Ponder toolkit for policy specification.

The prototype allows Goal Refinement considering the QoS-oriented high-level goals and the policy hierarchy. Figure 7-25 shows two out of the four generic goal graphs elaborated by the developer in the scenario.

1. The QoS Provisioning Management goal (top) has been refined into the six high-level goals for QoS Management. Then, the goal graph is built upon six sub-trees. Subsequently, the developer documents the relationship of these

high-level goals with the policies that control the corresponding high-level goals.

2. The Service Management Configured goal (goal graph in the bottom-right) has been refined considering the policy hierarchy, in this particular case, the Service Management Policy. The result of this process is a goal graph representing the different strategies for the Service Management function. Similar goal-graphs have been defined for the Traffic Engineering functions.

At service operation time the Administrator Consultant browses through the goal data base to carry out a Goal Selection process. For instance, for the high-level goal *Number of Subscriptions Controlled* in Figure 7-24b, the system guides the consultant to the *Subscription Logic Configured* goal, and subsequently to the *Service Subscription Configured* goal, the latter included in the Service Management Configured goal graph. At this moment of the selection, the consultant should decide the guidelines that reflect the administrative view about the *Service Subscription Configured* goal. For these, the "Conservative Settings" for both, the Satisfaction Level and the Admission Control, reflect the view of "Maximize subscriptions with the Highest Confidence to provide the agreed QoS" (selection marked with dotted lines in Figure 7-25). Similar selections should be achieved for the remaining "particular" goals of Figure 7-24b.

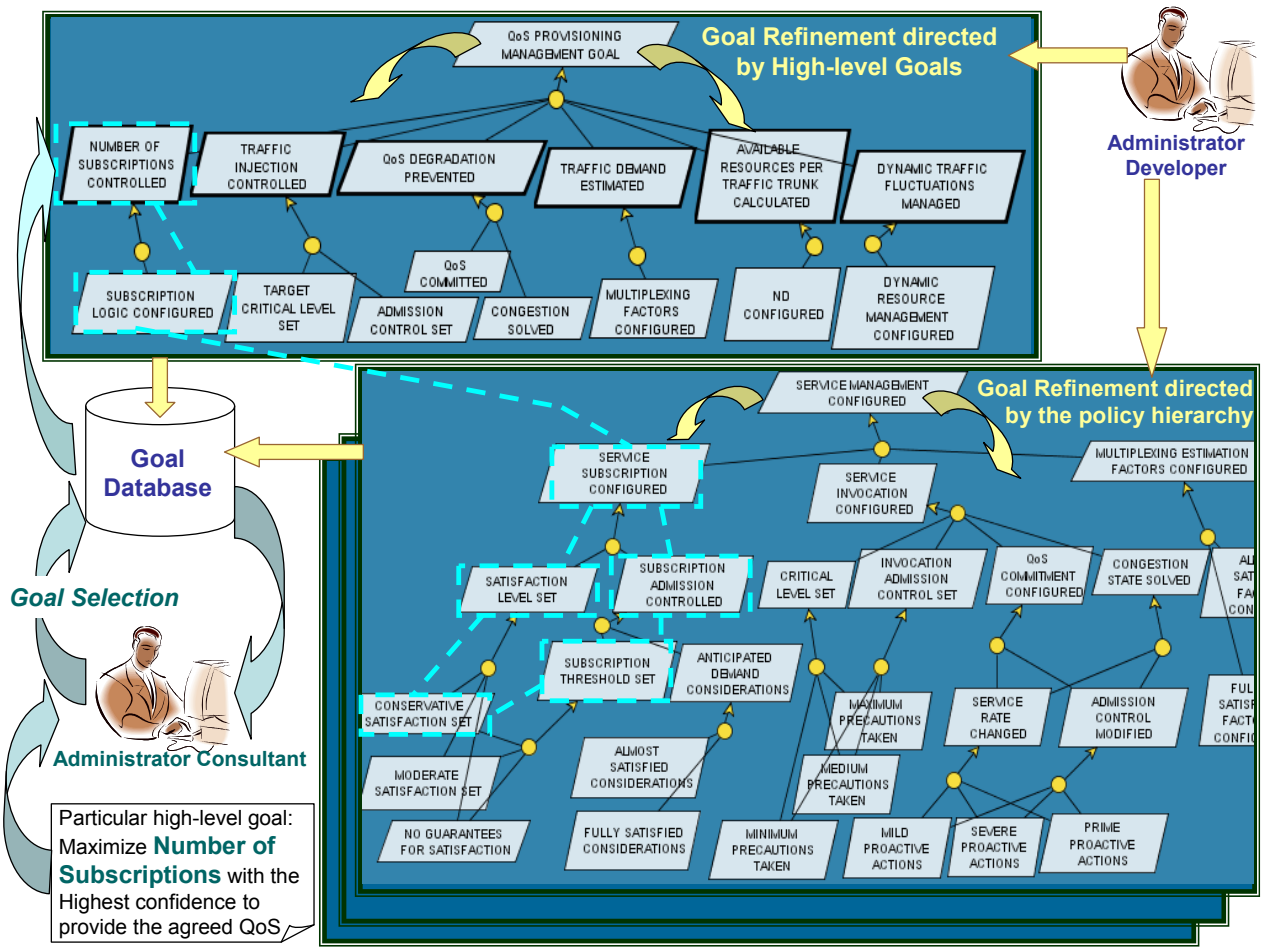


Figure 7-25. Practical aspects of the policy refinement process; goal refinement and goal selection.

Once the operative view for QoS provisioning has been defined, the resulting goal selection is verified for correctness and completeness. Later, the policy refinement

mechanisms produce the lowest-level executable policies that commit with such selection and consequently with the view of QoS Management.

In this scenario, five ND Policies, four TF Policies, two SLS-S Policies, six SLS-I Policies, and eight DRsM Policies have been produced in less than 3 seconds. Two of these lowest-level executable policies are shown in Figure 7-26.

Source | Code | Meta Policies

```
1  inst oblig /Managers/ConservativeSatisfaction {
2  on SRFactrSet ( serviceType ) ;
3  subject s = /Managers/SSM/SLSsPMA ;
4  target t = /Managers/SSM/SLSsPMA ;
5  do setSL ( serviceType . PHB , serviceType . SL = "1" ) ;
6  }
7  inst oblig /Managers/SubscriptionAcceptance {
8  on ramRecvd ( TT ) ;
9  subject s = /Managers/SSM/SLSsPMA ;
10 target t = /SLSs/MOs/bufferMO ;
11 do setMaxAccptConsrv ( TT ) ;
12 when t . getTrafType ( TT , TT . PHB )
13     -> t . getValue ( t . PHB , TT . SL ) > 0 ;
14 }
```

Figure 7-26. Sub-set of refined lowest-level policies in our scenario.

These are related to the Number of Subscriptions Controlled high-level goal. The ConservativeSatisfaction policy defines appropriate values on which the service admission mechanisms would rely; namely, the Satisfaction Level. The value assigned to the latter suggests that the active SLSs would enjoy their QoS at their almost satisfied rates even at congestion. Other policies have been produced to influence the parameters to define the almost satisfied rates. The SubscriptionAcceptance policy sets thresholds for incoming subscription requests. It is worth noticing that the events, subjects, targets, actions and constraints of the 25 lowest-level policies of our scenario have been abstracted automatically by the policy refinement mechanisms.

7.4.3.2.4 Summary

The model-checking approach provides an alternative mechanism for deriving sequences of actions that lead to the achievements of the goals and that can be encoded in Ponder obligation policies in the form of event condition action rules. This technique takes advantage from the recent results on the scalability of model checking techniques and is particularly suitable for temporal and concurrent systems behaviour. Tool support is available in order to support the methodology described.

7.5 Inter-domain Policy Aspects

The QoS management examples described in previous sections apply to individual administrative domains and more specifically to the management of traffic that initiates and terminates within the boundaries of a single domain. As such, the policies that drive the autonomic behaviour of ASs have a limited scope applying only to the treatment of local traffic. Challenges arise when services require traversing multiple independent domains from source to destination and, at the same time, maintaining the required quality. The QoS policy management solutions proposed in previous sections cease to have control over a service once this leaves the routing area of a Network Provider's (NP's) domain. For this reason, a new approach is required which can manage the key aspects of inter-domain QoS through policies.

7.5.1 Inter-domain QoS Management and Policies

Despite the complexities involved in managing QoS offered to local applications within a single domain, the task is simplified as Network Providers (NPs) have complete control over the resources utilised by those applications. In an inter-domain scenario a provider, although not possible to have control over external resources, would require some assurance about the quality of a service once out of its administrative domain. This can be achieved through negotiations between domains (peers) that can result in successful service level agreements referred to as provider SLSs (pSLSes).

Two main approaches have been proposed for the peering process between NPs: the source-based and the cascaded models [80], [81]. The source-based model follows a centralised approach by which the originating domain is aware of the end-to-end topology of the Internet and establishes pSLSes with a set of adjacent and distant domains in order to reach a destination, with a particular QoS. In the cascaded approach there are no explicit end-to-end agreements, with each NP building upon the capabilities of adjacent downstream domains to reach desired destinations. Figure 7-27 depicts this model where NP1 initiates a peering process aiming to reach a content server (CS) attached to NP3. The cascaded model is generally more accepted than the source-based one as the latter requires every originating domain to maintain an up-to-date topology of the Internet, making it a non-scalable solution.

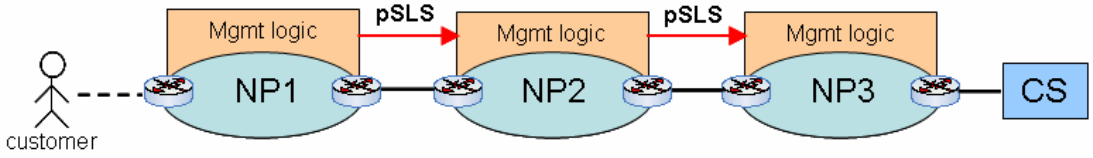


Figure 7-27. The cascaded model for setting up the end-to-end QoS chain.

The peering process consists of two phases where pSLSs are initially subscribed (pSLS-S) and then invoked (pSLS-I) during the activation of relevant services. Aspects of both phases can be achieved through policies which mainly manage the admission control process. During subscription, policies at the requesting provider can be used to decide which domains to negotiate with, based on their advertised capabilities and also to avoid specific NPs that are either too expensive or have a bad reputation in delivering QoS. pSLS-S policies at downstream providers (e.g. NP2) can guide the decision whether to accept, reject, or counter-offer a request based on the capabilities of their network in terms of QoS metrics, current resource availability and predicted network load. Once a pSLS has been successfully established along the delivery chain, policy-enriched invocation logic at each provider defines how much of the reserved BW, if not all, to allocate for that pSLS during the activation phase. In addition, dynamic policies at the invocation level aim to regulate the amount of pSLS traffic entering a domain so that it is in accordance with the agreed service rates thus preventing possible QoS deterioration of other services supported in a network.

As in the case of intra-domain traffic engineering, providers may opt for multiple inter-domain routes (Figure 7-28) to support a specific service for the purposes of load balancing and resilience to link failure. This can be achieved by establishing multiple pSLSs, where inter-domain route management policies can be used to define the number of alternative routes for a particular service and also the amount of load to assign to each of the routes.

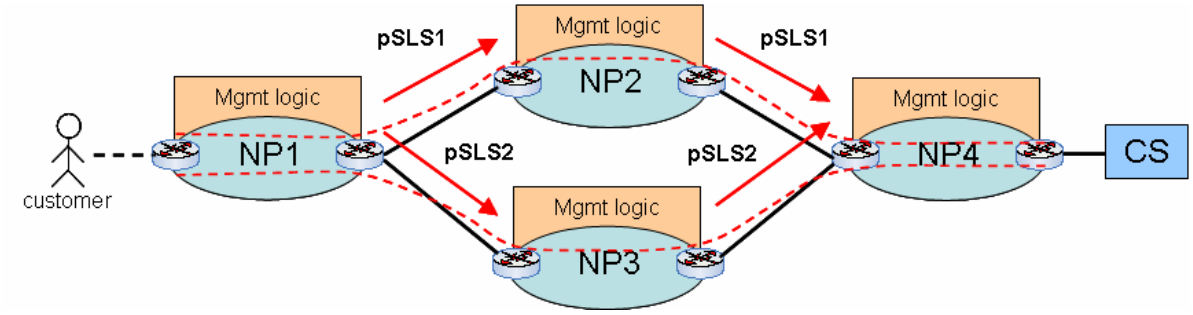


Figure 7-28. Creation of multiple pSLSs for a specific service.

The establishment of agreements between NPs commits each of the involved parties to support their contracts in terms of QoS characteristics and BW. This is realised through internal resource and route management functions in each domain which are driven by local policies. The latter define how transit traffic should be treated such that the desired pSLS QoS is delivered and at the same time the quality of local services is not compromised. Policy driven functions include resource optimisation through load balancing and dynamic resource management that take into account constraints such as maximum link utilisation per traffic type. In MPLS enabled networks deployed policies can result in the setup of explicit paths through parts of the network dedicated for inter-domain traffic but in cases that link sharing with local traffic is inevitable, resource management policies can be used to guide the distribution of capacity among traffic types.

7.5.2 QoS Management Policy Conflict Analysis in Collaborative Environments

Until now, provisioning has mostly been performed by each provider individually without taking into account the volume of traffic originating from other domains and the associated QoS requirements. This may result in suboptimal provisioning of a set of adjacent domains where the quality of both internal and external services can suffer from high packet drop rates and long delays. Establishing collaborations between NPs based on formal service level agreements can effectively result in a close-to-optimal end-to-end configuration, thus improving the quality that services enjoy.

In view of the fact that each provider maintains its own set of QoS management policies, the peering process is not a simple task as the objectives of each provider's policies may be conflicting between them in the sense that their coordinated effects can result in a sub-optimal inter-network operation. As such, negotiating providers, apart from exchanging information about traffic demand and QoS requirements, they should agree upon a "common denominator" at the policy level so that collaborative operation and provisioning is achieved. This involves identifying potential inconsistencies between inter-domain policies and applying reasoning techniques to resolve them by either modifying the local policies of one or both of the involved parties. One such example is the definition of the preferred outbound and inbound traffic exit and entry points. Consider the topology of Figure 7-29 where inter-domain links connect NP1 node egress1 (eg1) with NP2 node ingress1 (in1), and NP1 node egress2 with NP2 node ingress2 (in2). Now consider the routing policies of the two providers as follows:

- NP1: "*route traffic with destination domain X out from eg1*"
- NP2: "*accept traffic with destination domain X from in2*"

These rules clearly contradict each other and if the two providers ignore each other's policies then any packets from NP1 with destination *domain X* will be dropped at the

ingress of NP2. An effective negotiation process should support conflict detection and resolution mechanisms to first identify such inconsistencies and then provide strategies for their handling. Determining which of the involved providers should relax its policies can be based on incentives (e.g. economical), connectivity constraints, but most importantly on the impact on the overall network configuration and related policies.

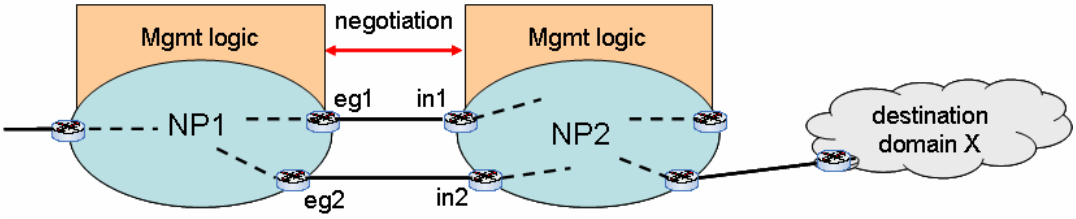


Figure 7-29. Ingress/egress point selection scenario.

Apart from conflicts related to policies managing inter-domain routing aspects, inconsistencies can potentially arise when negotiating BW and QoS parameters. These relate to policies of individual domains that determine the treatment of transit traffic in terms of allocation as well as delay and loss characteristics. Successful collaboration in setting up an end-to-end QoS chain can only be realised by resolving incompatibilities between the objectives of individual providers. This leads to a further issue that has to do with policies managing local and transit traffic within a domain. As mentioned in section 7.5.1, providers can opt for separate policies to accomplish this task, which essentially define internal routing operations and rules for sharing resources. Conflict analysis mechanisms can ensure the harmonic co-existence of these policies such that the quality of local applications is not compromised and at the same time peering contracts are not violated.

The issues described in this section are part of ongoing work in the area of inter-domain policy management that builds on the principles presented in [68] and [82].

7.6 Conclusions

The presented refinement methodology, techniques and tool support are a significant advancement upon the State of the Art and further enhance the potential of policy based paradigms for supporting Autonomic Management. Policy refinement adds a degree of automation to the system and simplifies administrators' tasks. Once the operative view for QoS provisioning has been defined, the resulting goal selection is verified for correctness and completeness. Later, the policy refinement mechanisms produce the lowest-level executable policies that commit with such selection and consequently with the view of QoS Management.

By addressing the issues of self-configuration and self-optimisation, policy refinement and policy based management in general are important steps towards Autonomic Management.

The refinement framework proposed overcomes the limitations of the KAOS goal elaboration method and in particular the lack of support to relate goal fulfilment to system's behaviour and consequently with the abstraction of policies that make possible such a goal fulfilment. For this purpose we have taken advantage of the logical foundations of the KAOS methodology and provided the formal underpinnings for the derivation of policies through both logic programming and model checking semantics. Up to now, these logical foundations have not been exploited in policy refinement contexts and consequently their potential to systematise the policy refinement process

have remained unexploited or unconsidered. Specifically, the innovations can be stated as follows:

- The use of specification patterns for behavioural properties as the means to represent the fulfilment of goals (temporal relationships amongst lowest-level goals). This has allowed us to represent goal fulfilment with formal notations and derive the policies that should be applied in order to achieve those goals through both abductive reasoning and model checking.
- The introduction of the Planning by Model Checking approach as the means to impose the behaviour that a managed system should exhibit as to fulfill High-level Goals. In addition, we have provided the methods that should be considered when adopting Model Checking searching engines, as the means to systematise the abstraction of policy-aware information from runtime system execution traces.
- The introduction of the abductive reasoning approach as the means of deriving the policies achieving the refined goals. This approach permits reasoning with partial information, incremental refinement, re-usable refinement through the encoding of the refinement results as new patterns and also permits policy analysis and validation e.g. for conflict detection as described in the next chapters.

Work done so far is open ended. We outline some of the most relevant directions that could be explored with regard to the framework presented in this work.

Our policy refinement process has been limited to consider exclusively Achieve goals, thus leading to relate the concept of achievable goals with obligation policies. Future work could be directed to explore the usefulness and implications of other goals supported by the KAOS methodology like Cease, Maintain and Avoid goals. We envisage that the consideration of these types of goals may enable to carry out analysis for application domains in which authorisation, access control and security issues have special relevance.

A key concept that has enabled us to systematise the policy refinement process is the Translation Process concept by means of which we derive policies from system state transitions. So far we have considered unconditional executions of actions, namely we have considered that all states and transitions are permitted in a policy based system. Future work could be directed to consider non permitted states likely dictated by Maintain/Avoid goals. In this sense, our Translation Process could be extended to consider the generation of actions based for example on guard conditions.

Finally, this refinement approach focuses on the derivation of the policies. Additional work is necessary in order to be able to automatically choose across multiple possible refinements based on non-functional requirements including performance and security.

8 Inter-domain Issues and Solutions

8.1 Introduction

As mentioned in section 1, most connectivity and higher level services in the Internet, relate to traffic that needs to span more than one domains in order to reach its destination. That means that, contrary to the case of intra-domain traffic, which is under

the exclusive control of a single domain (a single administrative entity) and is, therefore, to a certain extent relatively straightforward to handle, inter-domain traffic requires the cooperation and collaboration between multiple domains in order to be handled efficiently, e.g. in order for QoS guarantees of any kind to be provided.

Recognizing the significance of efficient handling of inter-domain traffic, related issues were identified and addressed in the context of EMANICS. To this end, the following four areas received research interest and effort:

- Joint intra and inter-domain traffic engineering
- Inter-domain resilience
- Inter-domain admission control
- Inter-domain policy based management

Joint intra and inter-domain traffic engineering aims to optimize jointly intra and inter-domain traffic so that a combined, overall optimized network performance can be achieved. Inter-domain resilience attempts to balance traffic in inter-domain links in order to achieve optimized performance during normal operations but also during failure conditions. The aim of inter-domain admission control is to efficiently manage inter-domain traffic, taking into account restrictions and challenges that a multi-domain environment raises, whereas inter-domain policy based management deals with the interactions and collaborations between independent providers, providing guidelines to achieve specific targets and resolve conflicts in such environments, where each provider may have its own objectives that can contradict with the other providers' ones.

In the rest of this section we will briefly describe the work performed in the context of EMANICS in the former three areas since inter-domain policy based management work has already been presented in section 7.5.

8.2 Joint Intra and Inter-domain Traffic Engineering

Traffic Engineering (TE) [83] is the set of techniques that optimize IP operational network performance by tactically routing traffic on a path other than the one would have been chosen if standard routing methods had been used. The task of TE is: given a network topology and a collected set of traffic demands (i.e. traffic matrix), determine the best routing for the traffic so that the overall network performance is optimized.

Today's Internet is a collection of more than 18,000 Autonomous Systems (ASes), each being an administrative region governed by its own network policies and routing protocols. Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (ISIS) are the common intra-AS routing protocols, while Border Gateway Protocol (BGP) is the de facto inter-AS routing protocol. With the hierarchical characteristics of the Internet, traffic is routed *within* an AS or *between* ASes. Thus, TE can be broadly divided into two types: intra-AS and inter-AS.

In intra-AS TE, the operator of an AS controls traffic routing within the network by either optimizing the link weights of the corresponding routing protocol (mostly OSPF or IS-IS) or establishing Label Switched Paths (LSPs) through MultiProtocol Label Switching (MPLS). Typical intra-AS TE optimization objectives are to minimize network bandwidth consumption and to achieve load balancing within the network. Inter-AS TE [10], [84], on the other hand, aims to control traffic entering and exiting an AS using optimization objectives such as load balancing over inter-AS links. It is commonly assumed that

these inter-AS links are frequently congestion points [6], [85]. For a particular AS, the network operator can control traffic exiting the AS by assigning the traffic to the 'best' egress points. This is called *Outbound* inter-AS TE. Likewise, the network operator can also control traffic entering the AS by selecting the 'best' ingress points, which is called *Inbound* inter-AS TE. In current practice, the commonly used method to enforce inter-AS TE is by adjusting BGP route attributes.

Internet Service Provider (ISP) networks typically carry both *intra-AS traffic* that is routed only within their networks and *inter-AS traffic* that is routed not only within their networks but also across other ASes. ISPs can employ both intra- and inter-AS TE to optimize the routing of these types of traffic. However, although some work exists on intra- and inter-AS TE, much of the existing literature deals with them *separately*: prior intra-AS TE work has assumed that ingress and egress points of inter-AS traffic do not change whereas prior inter-AS TE work has not considered route optimization within an AS. This is often inappropriate and results in suboptimal overall network performance due to the following two interaction effects between intra- and inter-AS TE:

- **Inter-on-Intra-AS TE**: The first interaction is the effect of inter-AS TE on the performance of intra-AS TE. Inter-AS TE can change the ingress and egress points of inter-AS traffic, thus causing the traffic to be routed on different ingress-to-egress paths within the network. This fundamentally changes the intra-AS traffic matrix, i.e. traffic load between each ingress and egress node pair. Such a change could therefore significantly affect the intra-AS TE solution.
- **Intra-on-Inter-AS TE**: The second interaction is the effect of intra-AS TE on the performance of inter-AS TE. It arises when outbound inter-AS TE and OSPF/ISIS-based intra-AS TE are used. One current practice is known as hot potato routing [86], [87]. In this approach, a BGP router will choose the closest exit as measured by the lowest IGP (Interior Gateway Protocol) cost among multiple equally-good egress points towards a downstream routing prefix. If the IGP costs are changed by intra-AS TE, some inter-AS traffic flows will in general be shifted to a new set of closest egress points. This could lead to congestion on these new egress points.

If intra- and inter-AS TE are not jointly optimized, a sequential approach may therefore be regarded as the current practice in the collective use of intra- and inter-AS TE. In this sequential approach, the solution of inter-AS TE becomes the input for intra-AS TE or vice versa. However, since the objectives and constraints of the subsequent stage are not taken into account, the decisions made at one stage often do not provide a good input for the subsequent one, sometimes even leading to infeasible inputs. As a result, it is difficult to claim that a truly good overall TE solution has been found when each TE is considered separately. We, therefore, propose to consider inter-AS TE during intra-AS TE and vice versa. In this work, we propose a joint optimization of intra- and inter-AS TE as an effective means to achieve better overall TE solutions than the one obtained by the sequential approach. Specifically, we investigate the following two challenges:

- *How should intra- and inter-AS TE be combined and how do we formulate their joint optimization?*
- *How can we solve the joint TE optimization problem to achieve a better overall network performance?*

Our contributions in this work are as follows. First of all, we explain with examples the two interaction effects that can lead to suboptimal overall network performance. Then, for the first challenge above, we formulate a bi-criteria joint optimization problem of intra- and inter-AS TE with an aim of optimizing their objectives simultaneously. Since the two interaction effects can generally apply to any intra- and inter-AS TE approach, it is possible to formulate the joint optimization problem for all the combinations of intra- and inter-AS TE approaches. However, as the primary objective of this work is to illustrate the point of interest on joint optimization of intra- and inter-AS TE in general, we only consider MPLS-based intra-AS TE and outbound inter-AS TE as an example in the joint optimization problem formulation. This problem formulation should be valid and practical as both intra-AS MPLS TE and outbound inter-AS TE are nowadays widely researched in academia and employed in industry. Moreover, a potential use of this problem formulation could be to optimize BGP/MPLS VPN provisioning, a subject which is currently attracting a great deal of industrial attention.

For the second challenge above, we consider three strategies to solve the joint TE optimization problem, namely sequential, nested and integrated optimization. These strategies aim to obtain non-dominated solutions with respect to the intra- and inter-AS TE objectives. We evaluate the performance of these strategies by simulation using Rocketfuel [88] topologies and synthetic traffic matrices. Our simulation results reveal that better overall network performance can be achieved by integrated optimization, solving intra- and inter-AS TE simultaneously. The performance improvement could allow the network to support a 30%-60% increase in the traffic demands. We believe that our work provides an insight into the interactions between intra- and inter-AS TE, enabling ISPs to further optimize the performances of their networks over the current practice sequential approaches.

For more details, the interested reader can refer to [89].

8.3 Inter-domain Resilience

8.3.1 Introduction

Outbound Traffic Engineering (TE) [10], [6] is a set of techniques for controlling traffic exiting a domain by assigning the traffic to the best egress points (i.e. routers or links). Outbound TE has become increasingly important and been well studied in the literature [6], [90], [91]. The general problem formulation of outbound TE is: given the network topology, BGP routing information and inter-domain Traffic Matrix (TM), determine the best Egress Point (EP) for each traffic demand so as to optimize the overall network performance [6]. Since inter-domain links are the most common bottlenecks in the Internet, optimizing their resource utilization becomes a key objective.

In practice, network conditions change dynamically, which can make fixed outbound TE solutions inaccurate and subsequently cause some inter-domain links to become congested over time. One such dynamic change is inter-domain traffic variation, which is typically caused by changes in user or application behaviour or even routing changes from other domains [92]. In addition to these traffic variations, transient and non-transient inter-domain peering link failure might occur. According to [94] transient inter-domain link failures are common events. Upon failure of a peering link, there may be a large amount of traffic shifted to other available EPs, potentially leading to congestion on these new serving EPs if they are not carefully determined. In theory, although it is possible to perform the outbound TE based on the other proposals in the literature [6], [90], [91] whenever any of those changes occur, it may require huge computational

overheads and a large number of EP re-configurations since previous proposals have not considered the reduction of reconfiguration changes and overheads. This can lead to excessive service disruptions and less practical to use. As a consequence, lack of TE solutions that react to those dynamic changes rapidly will leave the network *unmanaged*. It is thus the focus of this work to make outbound TE more adaptive to fast-changing IP networks by taking into consideration practical network operation and management constraints such as time-efficiency, reconfiguration overheads and service disruptions.

In this work, we propose an Inter-domain Outbound Traffic Engineering (IOTE) framework that consists of two re-optimization modules: a) Primary Egress Point (PEP) re-optimizer that is designed to manage dynamic traffic variation and routing changes. This module handles primary outbound TE which determines EP selection under Normal State (NS, i.e. no inter-domain link failure), and b) Secondary Egress Point (SEP) re-optimizer that is designed to manage inter-domain link failure. This module handles secondary outbound TE which determines EP selection under Failure States (FS, i.e. transient and non-transient inter-domain link failure). A time-efficient heuristic algorithm is proposed for each optimization module. The overall objective of the IOTE framework is, *in spite of dynamic changes in network conditions, to balance the loads among inter-domain links under both NS and FSs, while reducing reconfiguration overheads and service disruptions*.

To the best of our knowledge, there is no such an integrated network management approach like the IOTE framework that addresses primary and secondary outbound TE simultaneously. The authors in [93] propose a multi-objective outbound inter-domain TE re-optimization that handles changes of the traffic pattern or routing failures with a minimal burden on BGP. However, they do not consider the network performance under transient inter-domain link failures. On the other hand, the authors in [95] propose an intra-domain TE solution that is robust to transient intra-domain link failures and argue that relying on reactive robust solutions may not be appropriate or even feasible, since quickly computing and deploying a new robust solution can be challenging especially in today's large networks. Consequently, they propose a proactive robust solution to achieve their intra-domain TE objective. In similar fashion here, changing EP configuration dynamically to avoid a transient failure may not be a practical solution since there is not sufficient time for network operators to configure their networks before recovering from the transient failure. Instead, in order to avoid human configuration and achieve fast recovery from inter-domain link failure, we pursue a proactive robust approach to manage the transient inter-domain link failure by the pre-computation of SEPs. The following sections describe the framework and results in more detail.

8.3.2 Inter-domain Outbound Traffic Engineering Framework

The proposed IOTE framework is illustrated in Figure 8-1. The key idea of the framework is to continuously monitor the network conditions and, if some optimization triggering policies are met, initiate the PEP and SEP re-optimization modules based on the latest network conditions. The PEP and SEP solutions are then finally configured in the network if some implementation policies are met. The framework is constituted by three function blocks and we explain in details each of them as follows:

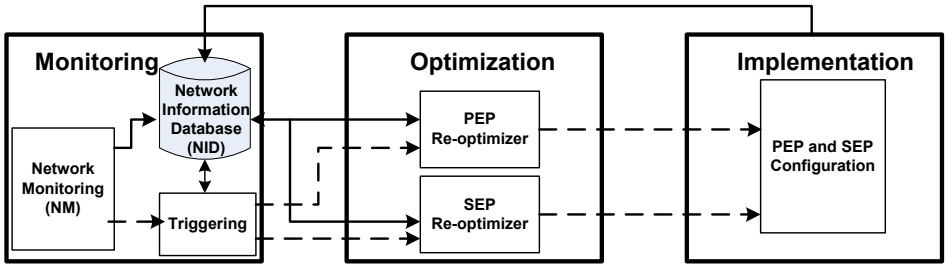


Figure 8-1. Inter-domain outbound traffic engineering framework.

Monitoring Block: it consists of Network Monitoring (NM) module, Network Information Database (NID) and triggering module. NM module continuously monitors the network conditions to establish a global view of the network information. The network information, which will be stored in the NID, includes inter-domain link utilization, overall traffic demands and BGP routing information. The authors in [96] presented a distributed management infrastructure that enables real-time views of network traffic to be generated. When a real-time global view of network is needed for network management, the console system that is controlled by the network operator retrieves and processes the information from the databases at each router through specific query languages. To apply this distributed monitoring infrastructure to outbound TE, each egress router monitors the utilization of inter-domain links attached to it and collects the updated BGP routing information from the local Routing Information Base (RIB). On the other hand, each ingress router monitors the updated traffic demands. Note that there are currently several hundred thousand prefixes in the Internet and collecting real-time changes for all the prefixes is, therefore, challenging. As suggested in [10], since TE can consider only a small number of prefixes that is responsible for large volume of traffic, this monitoring block only needs to pay attention to these prefixes in order to significantly reduce the monitoring complexity.

When the latest network conditions are known from monitoring, the NM module signals the triggering module. The triggering module invokes the re-optimization modules in the optimization block if some optimization triggering policies are met. In fact, the PEP and SEP re-optimizers are invoked if the latest maximum network utilization obtained by the monitoring exceeds a tolerance threshold α. This is a common policy since network providers often take actions to minimize congestion in their networks. Without loss of generality, in this work we assume α =50% to be the borderline of congestion. In summary, the PEP/SEP re-optimizers aim to keep the network utilization under NS and FSs below α.

Optimization Block: it consists of PEP and SEP re-optimizers and requires as input the latest network information from the NID. The task of PEP re-optimizer is to re-assign the primary egress points to traffic flows under NS. The key objective is to achieve inter-domain load balancing while reducing reconfiguration overheads and service disruptions. The PEP re-optimizer is designed for managing dynamic traffic variation and routing changes. On the other hand, the task of SEP re-optimizer is to pre-compute a set of optimal secondary (i.e. backup) egress points for the traffic. Upon failure of an inter-domain link, the traffic affected by the failure will be shifted to the secondary egress points. The key objective is to achieve inter-domain load balance under any single inter-domain link failure while reducing backup reconfigurations. The SEP re-optimizer is designed for managing inter-domain link failure. It is worth to mention that

changing secondary egress points does not cause any service disruption since the primary BGP routes are remain intact.

Implementation Block: The implementation block enforces the solutions produced by the PEP and SEP re-optimizer into the network based on some performance policies. A benefit-based performance policy can be applied as follows. The PEP and SEP solutions are enforced if there is a gain in reducing the worst-case Maximum Link Utilization (MLU) compared to the current attempt (i.e. current solution implemented in the network). Note that there is a trade off between the gain that can be obtained by reducing the egress point utilization and up-to-date PEP and SEP configuration. If a large gain is chosen, not many re-optimization solutions can satisfy the required gain. This leads to less frequent PEP and SEP reconfiguration (i.e. increasing the lifetime of the current solution) and the PEP and SEP configuration tend to become obsolete. This results in a less load balanced network especially in the presence of failures. Hence, the choice of gain for solution implementation depends on how often the network operators are willing to change the network configuration and how evenly balanced they want their network to be.

8.3.3 Evaluation Results

We compare the performance of IOTE framework with two alternative strategies. The first strategy, No-Reopt, does not consider any PEP or SEP re-optimization at all, while the second, PEP-Reopt-Only only considers PEP re-optimization.

In this section we show our results evaluation on a topology with 20 EPs. We assume equal EP capacities and consider their value to be OC-48 (2.5Gbps). In line with [6], we consider 1000 popular destination prefixes. In fact, each of them may not merely represent an individual prefix but also a group of distinct destination prefixes that have the same set of candidate EPs, in order to improve network and TE algorithm scalability. Without loss of generality, we consider Half Prefix Reachability (HPR) which means that each prefix is reachable through only half of the total EPs. According to [97], the volume of inter-AS traffic demand is top-heavy and it can be approximated by a Weibull distribution with the shape parameter equal to 0.2-0.3. We therefore generate the inter-AS TM following this distribution with the shape parameter equal to 0.3.

Figure 8-2 shows the set of randomly generated events used for our evaluation. These events include traffic variation, routing changes, transient and non-transient inter-domain link failure.

Figure 8-3, Figure 8-4 and Figure 8-5 show that during the first interval all the strategies perform identical by both under NS and FS. This is due to our assumption for fair comparison that all the strategies start with the same initial solutions. However, once their monitored performance violates the re-optimization triggering threshold value (i.e. 50%), they start to react differently.

Figure 8-3 shows that the No-Reopt is the worst performer under all the events and not only cannot keep the inter-AS MLU under NS below the threshold value but also its MLU under FSs has dramatically poor performance. This phenomenon was expected due to the fact that this strategy does not perform any re-optimization to achieve load balancing. As a result, its initial PEP and SEP solutions become vulnerable to the subsequent changes in the network conditions such as accumulation of traffic matrix variations and routing changes.

In contrast, Figure 8-4 shows that the PEP-Reopt-Only can keep inter-AS MLU only under NS below the threshold value. However, since this strategy ignores SEP re-optimization, its MLU under FSs becomes poor and gets worse after subsequent events. Nevertheless, the overall FS network performance degradation in the PEP-Reopt-Only is less severe than in No-Reopt. This result is expected since the No-Reopt does not apply any re-optimization as a result the failure of a congested EP and the assignment of its traffic flows over the non-optimized SEP may result in the re-assignment of a large number of traffic flows over already congested EPs, causing significant performance degradation. On the contrary, in the PEP-Reopt-Only, an EP failure and the re-assignment of its flows over the non-optimized SEP does not lead to much performance degradation due to the fact that the EPs are balanced under NS by PEP re-optimization. However, Figure 8-5 shows that IOTE Framework can keep MLU not only under NS but also under most of FSs below the threshold value by re-optimizations.

		RC	STD	STI	STI	RC	STD	STI	STD
2 TF	3 TF	1 TF 1 NTF	2 TF	3 TF	1 TF 1 NTF	2 TF	1 TF 1 NTF	2 TF	2 TF
STF	GTI	STF	GTI	GTD	STF	GTD	STF	GTI	STF

Figure 8-2. Set of randomly generated events for 20-EP topology.

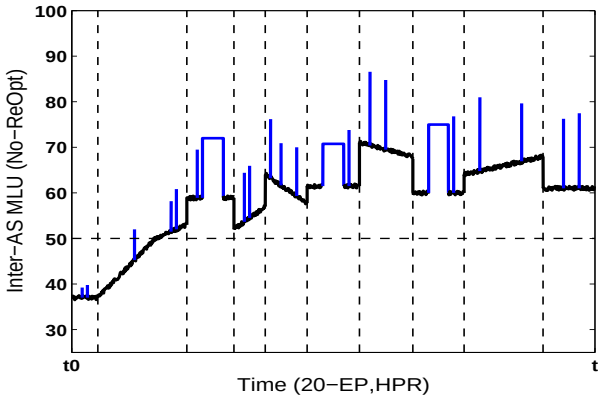


Figure 8-3. Inter-AS MLU performance of No-Reopt for 20-EP

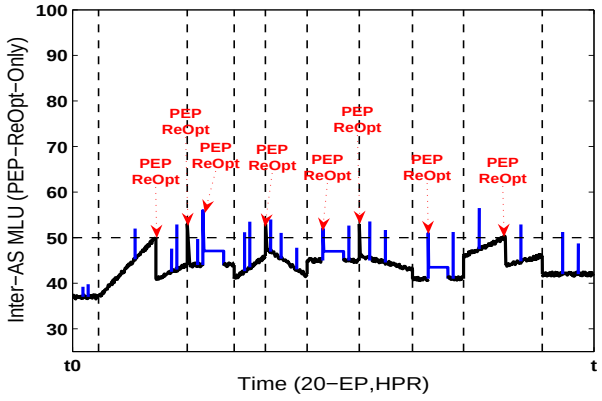


Figure 8-4. Inter-AS MLU performance of PEP-Reopt-Only for 20-EP.

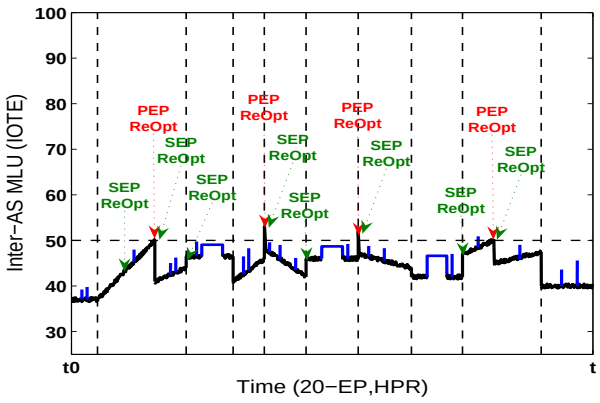


Figure 8-5. Inter-AS MLU performance of IOTE Framework for 20-EP.

On the whole, our proposed IOTE Framework has achieved: a) much better performance in terms of the inter-AS MLU under NS compared to the No-Reopt (i.e. its worst-case NS performance is 30% better) and almost the same performance as the PEP-Reopt-Only, and b) significantly better performance in terms of the inter-AS MLU under FSs compared to No-Reopt (i.e. its worst-case FS performance is 41% better) and better performance compared to PEP-Reopt-Only (its worst-case FS performance is 10% better). Also to achieve these results, the IOTE Framework incurs almost 58% less service disruptions and 46% less PEP reconfigurations compared to the PEP-Reopt-Only at the cost of 1425 SEP reconfigurations. We recall that the SEP reconfiguration does not cause service disruption. In addition, less service disruptions and PEP reconfigurations in our framework may imply better network stability compared to the PEP-Reopt-Only.

For more details the interested reader can refer to [9].

8.4 Inter-domain Admission Control

8.4.1 Overview of Work in the Area of Admission Control

The role of admission control is to control the amount of traffic injected into a network so as to prevent overload situations that can lead to QoS violations. Significant research effort has been given in this area and numerous admission control schemes appear in the literature. The various admission control schemes can be classified into three categories: a) endpoint admission control (EAC), b) traffic descriptor-based admission control (TDAC), and c) measurement-based admission control (MBAC).

EAC is based on metrics applied to probing packets sent along the transmission path before the flow is established [98]. The probing packets can be sent either at the same priority as flow packets (in-band probing) or at a lower priority (out-of-band probing). One problem of EAC schemes is that simultaneous probing by many sources can lead to a situation known as thrashing. That is, even though the number of admitted flows is small, the cumulative level of probing packets prevents further admissions.

TDAC is based on the assumption that traffic descriptors are provided for each flow prior to its establishment. This approach achieves high utilization when the traffic descriptors used by the scheme are appropriate. Nevertheless, in practice, it suffers from several problems [99]; the most important being the inability to come up with appropriate traffic descriptors before establishing the flow.

MBAC tries to avoid this problem by shifting the task of traffic characterization to the network [99]. That means that the network attempts to “learn” the characteristics of existing flows through real-time measurements. This approach has certain advantages. For example, a conservative specification does not result in overallocation of resources for the entire duration of the service session. Also, when traffic from different flows is multiplexed, the QoS experienced depends on their aggregate behaviour, the statistics of which are easier to estimate. However, relying on measured quantities raises issues, such as estimation errors and memory related issues.

Most of the schemes, to be applicable in practice, explicitly or implicitly make the assumption that the traffic is intra-domain; that is it originates and terminates within the same domain. For example, EAC schemes that require intermediate routers involvement when processing the probe packets, are not suitable for inter-domain traffic since router operations in other domains cannot be forced.

The schemes that do not make this assumption, in many cases, e.g. see [100], require the cooperation of adjacent domains along the end-to-end paths on a per-flow basis (scalability issues) as well as the existence of a commonly understandable signalling protocol end-to-end in order to perform admission control in each domain and propagate downstream the admission control decision and/or the QoS received so far (compatibility issues).

Contrary to these schemes, in our work we present a measurement-based admission control scheme for inter-domain traffic, which when deployed in the context of a cascaded QoS peering model, does not require the cooperation and signalling among adjacent domains on a per-flow basis. In the next section we will briefly describe our admission control scheme (we call it inter-MBAC).

8.4.2 Overview of inter-MBAC

The practical assumption of our scheme is that a QoS peering model similar to the one proposed by the MESCAL project [101] exists. In that scheme, domains negotiate and agree pSLAs with immediately interconnected domains in order to be able to inter-domain QoS. Figure 8-6 shows the operations in such peering model.

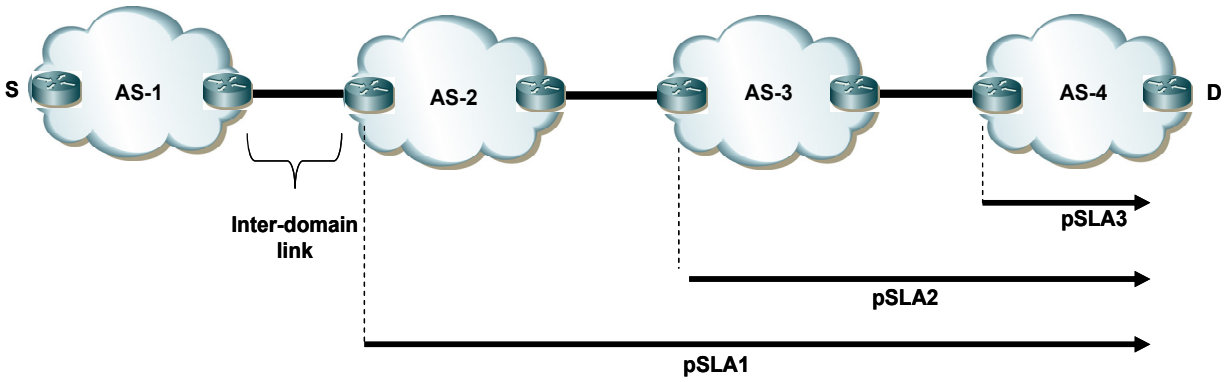


Figure 8-6. Example of a QoS peering model.

In Figure 8-6, AS-3 establishes a pSLA ($pSLA_3$) with AS-4 that allows it to reach destinations D. Then by combining its local QoS capabilities with the QoS agreed in $pSLA_3$, AS-3 is able to offer $pSLA_2$ to AS-2 that will allow traffic from AS-2 to reach destinations D. In a similar way, AS-2 can offer to AS-1, $pSLA_1$ so that traffic from sources S of AS-1 can reach destinations D. This model follows the current practise of peering agreements between providers and is BGP compatible.

The benefits of this approach is that providers can deploy their own admission control approaches within their own domains to achieve 'local' QoS objectives, which when combined with the QoS capabilities in the pSLAs, they can provide end-to-end QoS. Therefore, no cooperation/signalling on a per-flow basis among domains is needed.

Inter-MBAC, in principle, attempts to guarantee this 'local' QoS by keeping per-flow state information only at the ingress nodes of the domain where it is applied, based on congestion status feedback information from the egress nodes of the domain. That is, it attempts to perform QoS 'balancing' between ingress and egress nodes so that, in total, the 'local' QoS target is achieved.

Inter-MBAC's performance and its ability to provide the requested 'local' QoS for a variety of traffic mix scenarios and loading conditions has been shown by means of simulations.

For more details regarding inter-MBAC's functionality and its performance, both absolute and relative compared to other existing admission control schemes, the interested reader can refer to [102].

9 Summary and Conclusions

The scope of this deliverable was to present the work performed in the area of autonomic management in the context of EMANICS. This work was the result of an ongoing collaborative effort between the EMANICS partners.

To this end, we first presented an approach for autonomic management and protection of ubiquitous environments and showed how this approach can ensure the proper identification and punishment of misbehaving devices in such an environment so that they can be prevented from causing general disruptions.

We also presented an automated integrated network management framework that addresses IT availability and optimizes the use of network resources during normal operation and also assists operators to generate recovery plans and actions accordingly, which reduce network re-configuration overhead and service disruptions.

We subsequently addressed the issue of context management and provisioning in ubiquitous environments and showed how context management architectures can be enhanced with components that follow the principles of autonomic management.

Distributed and autonomic real-time traffic analysis was also a topic covered in this deliverable and we showed that the solution proposed allows discovery of available processing and storage resources for traffic analysis and flow collection in a fully distributed and autonomic manner with all the benefits that this allows for.

Policy based approaches for autonomic intra and inter-domain network management were also investigated and we showed how they can effectively address the issues of self-configuration and self-optimization as two of the key properties of autonomic systems. Realistic case studies demonstrated the applicability of the policy based management techniques in realistic scenarios.

Finally, we presented inter-domain related issues identified in EMANICS and work that has been performed to address these issues.

The work presented in this deliverable proves that collaboration links among all partners of WP9 were built. Future joint activities based on the work presented in this deliverable have been planned and are being refined.

10 References

[1] D. Verma, “Simplifying network administration using policy based management,” IEEE Network, vol.16, issue 2, Mar-Apr 2002.

[2] A. Hadjiantonis, A. Malatras and G. Pavlou, “A context-aware, policy based framework for the management of MANETs”, IEEE Policy 2006.

[3] H. Yang, J. Shu, X. Meng and S. Lu, “SCAN: Self-organized network-layer security in mobile ad hoc networks”, IEEE Journal on Selected Areas in Communications, vol. 24, issue 2, February 2006.

[4] J. Hubaux, L. Buttyán, and S. Čapkun, “The quest for security in mobile ad hoc networks”, 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2001.

[5] K. Bradley, S. Cheung, N. Puketza, B. Mukherjee and R. Olsson, “Detecting disruptive routers: a distributed network monitoring approach,” 1998 Symposium on Security and Privacy.

[6] T. Bressoud, R. Rastogi and M. Smith, “Optimal configuration for bgp route selection”, IEEE INFOCOM 2003.

[7] A. Aamodt and E. Plaza, “Case-based reasoning: foundational issues, methodological variations and system approaches” Artificial Intelligence Communications, vol. 7, issue 1, 1994.

[8] H. Tran and J. Schönwälder, “Distributed case-based reasoning for fault management”, 1st International Conference on Autonomous Infrastructure, Management and Security, 2007.

[9] M. Amin, K. Ho, M. Howarth and G. Pavlou, “An integrated network management framework for inter-domain outbound traffic engineering”, IEEE/IFIP MMNS 2006.

[10] N. Feamster, J. Borkenhagen and J. Rexford, “Guidelines for inter-domain traffic engineering”, ACM SIGCOMM Computer Communications Review, vol. 33, issue 5, October 2003.

[11] H. Tran and J. Schönwälder, “Heuristic search using a feedback scheme in unstructured peer-to-peer networks”, 5th International Workshop on Databases, Information Systems and P2P Computing, 2007.

[12] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld and D. Wilkins, “Pddl - the planning domain definition language”, Yale Center for Computational Vision and Control Tech Report CVC TR-98-003/DCS TR-1165, October 1998.

[13] M. Ghallab, D. Nau and P. Traverso, “Automated planning - theory and practice”, Elsevier, 2004.

[14] M. Brodie, S. Ma, G. Lohman, T. Syeda-Mahmood, L. Mignet, N. Modani, J. Champlin and P. Sohn, “Quickly finding known software problems via automated symptom matching”, 2nd International Conference on Automatic Computing, 2005.

[15] S. Montani and C. Anglano, “Case-based reasoning for autonomous service failure diagnosis and remediation in software systems”, 8th European Conference on Case-Based Reasoning, 2006.

[16] A. Hadjiantonis, M. Charalambides and G. Pavlou, “A policy based approach for managing ubiquitous networks in urban spaces”, IEEE ICC 2007.

[17] P. Flegkas, P. Trimintzios and G. Pavlou, “A policy based quality of service management system for ip diffserv networks”, IEEE Network, vol. 16, issue 2, Mar-Apr 2002.

[18] J. Kephart, “Research challenges of autonomic computing”, 27th International Conference on Software Engineering, 2005.

[19] B. Srivastava and S. Kambhampati, “The case for automated planning in autonomic computing”, IEEE ICAC 2005.

[20] N. Arshad, D. Heimbigner and A. Wolf, “A planning based approach to failure recovery in distributed systems”, 1st ACM SIGSOFT Workshop on Self-managed Systems, 2004.

[21] J. Kephart and D. Chess, "The vision of autonomic computing", ACM Computer, vol. 36, issue 1, January 2003.

[22] K. Curran, M. Mulvenna, A. Galis and C. Nugent, “Challenges and research directions in autonomic communications”, International Journal of Internet Protocol Technology, vol. 2, issue 1, January 2007.

[23] P. Gray and D. Salber, “Modeling and using sensed context information in the design of interactive applications”, 8th IFIP Working Conference on Engineering for Human-Computer Interaction, 2001.

[24] K. Henricksen, J. Indulska and A. Rakotonirainy, “Modeling context information in pervasive computing systems”, 1st International Conference on Pervasive Computing, 2002.

[25] Held, S. Buchholz and A. Schill, “Modeling of context information for pervasive computing applications”, 6th World Multiconference on Systemics, Cybernetics and Informatics, 2002.

[26] T. Strang, C. Linnhoff-Popien and F. Korbinian, “CoOL: A context ontology language to enable contextual interoperability”, 4th International Conference on Distributed Applications and Interoperable Systems, 2003.

[27] T. Gu, X. Wang, H. Pung and D. Zhang, “An ontology-based context model in intelligent environments”, Communication Networks and Distributed Systems Modeling and Simulation Conference, 2004.

[28] H. Chen, F. Perich, T. Finin, and A. Joshi, “SOUPA: Standard ontology for ubiquitous and pervasive applications”, 1st Annual International Conference on Mobile and Ubiquitous Computing, 2004.

[29] K. Henricksen, J. Indulska and T. McFadden, “Modeling context information with ORM,” On the Move to Meaningful Internet Systems 2005: OTM Workshops, LNCS 3762, 2005.

[30] S. Davy, K. Barrett, M. Serrano, J. Strassner, B. Jennings and S. van der Meer, “Policy interactions and management of traffic engineering services based on ontologies”, Latin American Network Operations and Management Symposium, 2007.

[31] A. Dey, “Providing architectural support for building context-aware applications”, PhD thesis, Georgia Institute of Technology, 2000.

[32] G. Judd, and P. Steenkiste, “Providing contextual information to pervasive computing applications”, 1st IEEE International Conference on Pervasive Computing and Communications, 2003.

[33] N. Samaan, H. Harroud and A. Karmouch, “PACMAN: A policy based architecture for context management in ambient networks”, 4th IEEE Consumer Communications and Networking Conference, 2007.

[34] M. Saternus, T. Weis, M. Knoll and F. Dürr, “A middleware for context-aware applications and services based on messenger protocols”, 5th Annual IEEE International Conference on Pervasive Computing and Communications, 2007.

[35] H. Chen, “An intelligent broker architecture for pervasive context-aware systems," PhD Thesis, University of Maryland, 2004.

[36] T. Buchholz, M. Krause, C. Linnhoff-Popien and M. Schiffers, “CoCo: Dynamic composition of context information”, 1st Annual International Conference on Mobile and Ubiquitous Computing, 2004.

[37] T. Strang and C. Linnhoff–Popien, “A context modeling survey,” Workshop on Advanced Context Modeling, Reasoning and Management, 2004.

[38] I. Hochstatter, M. Duergner and M. Krause, “A context middleware using an ontology–based information model”, Dependable and Adaptable Networks and Services, LNCS 4606, 2007.

[39] M. Serrano, J. Serrat, J. Strassner and M. Ó Foghlú, “Management and context integration based on ontologies, behind the interoperability in autonomic communications”, System and Information Sciences Notes, vol. 1, issue 4, April 2007.

[40] M. Serrano, J. Serrat and J. Strassner, “Ontology-based reasoning for supporting context-aware services on autonomic networks”, IEEE ICC 2007.

[41] De Bruijn, R. Lara, A. Polleres and D. Fensel, “OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web”, 14th International World Wide Web Conference, 2005.

[42] A. Ganek and T. Corbi, “The dawning of the autonomic computing era”, IBM Systems Journal, vol. 42, issue 1, January 2003.

[43] M. Serrano, J. Serrat and A. Galis, “Ontology-based context information modelling for managing pervasive applications”, International Conference on Autonomic and Autonomous Systems, 2006.

[44] J. Strassner and D. Raymer, “Implementing next generation services using policy based management and autonomic computing principles,” IEEE/IFIP NOMS 2006.

[45] J. Strassner, “Seamless mobility – A compelling blend of ubiquitous computing and autonomic computing”, Dagstuhl Workshop on Autonomic Networking, 2006.

[46] B. Claise, “Cisco Systems NetFlow Services Export Version 9”, Internet RFC 3954, October 2004.

[47] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May and A. Lakhina, “Impact of packet sampling on anomaly detection metrics”, 6th ACM SIGCOMM Conference on Internet Measurement, 2006.

[48] K. Claffy, G. Polyzos and H. Braun, “Application of sampling methodologies to network traffic characterization”, ACM SIGCOMM 1993.

[49] W. Cochran, “Sampling Techniques”, Wiley, 1987.

[50] N. Duffield and M. Grossglauser, “Trajectory sampling for direct traffic observation”, IEEE/ACM Transactions on Networking, vol. 9, issue 3, June 2001.

[51] N. Duffield, C. Lund and M. Thorup, “Learn more, sample less: control of volume and variance in network measurement”, IEEE Transactions of Information Theory, vol. 51, issue 5, May 2005.

[52] J. Mai, C. Chuah, A. Sridharan and H.Ye, “Is sampled data sufficient for anomaly detection?”, 6th ACM SIGCOMM Conference on Internet Measurement, 2006.

[53] C. Morariu and Burkhard Stiller, “DiCAP: Distributed packet capturing architecture for high-speed Network Links”, IFI Technical Report, No. <to-be-received>, June 2008.

[54] C. Morariu, T. Kramis and Burkhard Stiller, “DIPStorage: Distributed architecture for storage of ip flow records”,16th Workshop on Local and Metropolitan Area Networks, 2008.

[55] NFDUMP Website: http://nfdump.sourceforge.net/.

[56] Flow-Tools Website: http://www.splintered.net/sw/flow-tools/.

[57] Postgreqsl website: http://www.postgresql.com/.

[58] MySQL website: http://www.mysql.com/.

[59] J. Moffet and M. Sloman, “Policy hierarchies for distributed systems management”, IEEE Journal on Selected Areas in Communications, vol. 11, issue 9, December 1993

[60] R. Darimont and A. van Lamsweerde, “Formal refinement patterns for goal-driven requirements elaboration”, 4th ACM Symposium on the Foundations of Software Engineering,1996.

[61] R. Darimont, E. Delor, P. Massonet and A. van Lamsweerde, “GRAIL/KAOS: an environment for goal-driven requirements engineering”, 20th International Conference on Software Engineering, 1998.

[62] F. Bacchus and F. Kabanza, “Using temporal logics to express search control knowledge for planning”, Artificial Intelligence Journal, vol. 116, issue 1-2, January 2000

[63] M. Dwyer, G. Avrunin and J. Corbett, “Property specification patterns for finite-state verification”, Workshop on Formal Methods in Software Practice, 1998

[64] M. Dwyer, G. Avrunin, and J. Corbett, “A system of specification patterns”. http://patterns.projects.cis.ksu.edu/

[65] A. van Lamsweerde, “Goal-oriented requirements engineering: a guided tour”, 5th IEEE International Symposium on Requirements Engineering, 2001.

[66] F. Bacchus and F. Kabanza, “Planning for temporally extended goals”, Annals of Mathematics and Artificial Intelligence, vol. 22, issue 1-2, 1998.

[67] M. Sloman, “Policy driven management for distributed systems”, Journal of Network and Systems Management, vol. 2, issue 4, 1994.

[68] A. Bandara, E. Lupu and A. Russo, “Using event calculus to formalise policy specification and analysis”, IEEE Policy 2003.

[69] B. van Nuffelen and A. Kakas, “A-System: programming with abduction”, 6th International Conference on Logic Programming and Non-monotonic Reasoning, 2001.

[70] N. Damianou, T. Tonouchi, N. Dulay, E. Lupu, and M. Sloman, “Tools for domain-based policy management of distributed systems”, IEEE/IFIP NOMS 2002.

[71] F. Giunchiglia and P. Traverso, “Planning as model checking”, 5th European Conference on Planning, 1999.

[72] A. Cimatti , M. Pistore , M. Roveri and P. Traverso, “Weak, strong, and strong cyclic planning via symbolic model checking” Artificial Intelligence, vol. 147, issue 1-2, July 2003.

[73] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou and A. Lafuente, “Using linear temporal model checking for goal-oriented policy refinement frameworks”, IEEE Policy 2005.

[74] F. Kabanza, M. Barbeau and R. St-Denis, “Planning control rules for reactive agents”, Artificial Intelligence, vol. 95, issue 1, August 1997.

[75] S. Thiébaux, C. Gretton, J. Slaney, D. Price and F. Kabanza, “Decision-theoretic planning with non-markovian rewards”, Journal of Artificial Intelligence Research, vol. 25, January 2006

[76] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas and G. Pavlou, “A functional solution for goal-oriented policy refinement”, IEEE Policy 2006.

[77] J. Strassner, “Policy based network management”, Elsevier, 2004

[78] N. Damianou, N. Dulay, E. Lupu and M. Sloman, “The ponder policy specification Language. Policy Workshop 2001.

[79] C. Bartolini, M. Sallé and D. Trastour, “IT service management driven by business objectives - an application to incident management”, IEEE/IFIP NOMS 2006.

[80] M. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morrand, H. Asgari and P. Georgatsos, “Provisioning for inter-domain quality of service: the MESCAL approach,” IEEE Communications Magazine, June 2005.

[81] P. Georgatsos J. Spencer, D. Griffin, T. Damilatis, H. Asgari, J. Griem, G. Pavlou and P. Morrand, “Provider-level service agreements for inter-domain QoS delivery”, Lecture Notes in Computer Science, vol. 3266, September 2004.

[82] M. Charalambides, P. Flegkas, G. Pavlou, A. Bandara, E. Lupu, A. Russo, N. Dulav, M. Sloman and J. Rubio-Loyola, “Policy conflict analysis for quality of service management”, IEEE Policy 2005.

[83] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, “Overview and principles of internet traffic engineering”, Internet RFC 3272, May 2002.

[84] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure and S. Uglih, “Inter-domain traffic engineering with bgp”, IEEE Communication Magazine, May 2003.

[85] N. Hu, L. Li, Z. Mao, P. Steenkiste and J. Wang, “Locating internet bottlenecks: algorithms, measurements and implications”, ACM SIGCOMM 2004.

[86] R. Teixeira, A. Saikh, T. Griffin and J. Rexford, “Dynamics of hot-potato routing in ip networks”, ACM SIGMETRICS 2004.

[87] R. Teixeira, A. Saikh, T. Griffin and J. Rexford, “Network sensitivity to hot-potato disruptions”, ACM SIGCOMM 2004.

[88] N. Spring, R. Majahan and D. Wetherall, “Measuring isp topologies with rocketfuel”, ACM SIGCOMM 2002.

[89] K. Ho, M. Howarth, N. Wang, G. Pavlou and S. Georgoulas, “Joint optimization of intra- and inter-autonomous systems traffic engineering”, IEEE/IFIP NOMS 2006.

[90] S. Uhlig, O. Bonaventure and B. Quotin, “Inter-domain traffic engineering with minimal bgp configurations”, 18th International Teletraffic Congress, 2003.

[91] K. Ho, N. Wang, P. Trimintzios and G. Pavlou, “Multi-objective egress router selection policies for inter-domain traffic with bandwidth guarantees”, IFIP Networking, 2004.

[92] R.Teixeira, N. Duffield, J. Rexford and M. Roughan, “Traffic matrix reloaded: impact of routing changes”, Passive and Active Network Measurements, 2005.

[93] S. Uhlig and O. Bonaventure, “Designing bgp-based outbound traffic engineering techniques for stub ASes”, ACM SIGCOMM Computer Communications Review, vol. 34, issue 5, October 2004.

[94] O. Bonaventure, C. Filsfils and P. Francois, “Achieving sub-50 milliseconds recovery upon bgp peering link failures,” ACM CONEXT, 2005.

[95] A. Sridharan and R. Guerin, “Making igp routing robust to link failures2”, IFIP Networking, 2005.

[96] K. Lim and R. Stadler, “Real-time views of network traffic using decentralized management”, IFIP/IEEE IM 2005.

[97] A. Broido, Y. Hyun, R. Gao and K. Claffy, “Their share: diversity and disparity in ip traffic”, Passive and Active Network Measurements, 2004.

[98] L. Breslau, E. Knightly, S. Shenker, I. Stoica and H. Zhang, “Endpoint admission control: architectural issues and performance”, ACM SIGCOMM 2000.

[99] M. Grossglauser and D. Tse, “A framework for robust measurement-based admission control”, IEEE/ACM Transactions on Networking, vol. 7, issue 3, June 1999.

[100] www.mescal.org.

[101] S. Lima, P. Carvalho and V. Freitas, “Distributed admission control for qos and sls management”, Journal of Network and Systems Management, vol 12, issue 3, September 2004.

[102] S. Georgoulas, G. Pavlou, P. Trimintzios and K. Ho, “Admission control for inter-domain real-time traffic originating from differentiated services stub domains”, WWIC 2007.

11 Abbreviations

AI Artificial Intelligence

API	Application Programming Interface
AS	Autonomous System
BGP	Border Gateway Protocol
CAS	Context-Aware Service
CIM	Common information Model
CIS	Context Information Service
CMM	Context Meta-Model
CoCo	Context Composition
DiffServ	Differentiated Services
DMTF	Distributed Management Task Force
FCAPS	Fault, Configuration, Accounting, Performance, and Security
EDPS	European Data Protection Supervisor
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
LBS	Location Based Services
LTL	Linear Temporal Logic
OSM	Ontologies for Support and Management
PBM	Policy Based Management
MANET	Mobile Ad Hoc Network
NREN	National Research and Education network
MPLS	Multiprotocol Label Switching
OWL	Web Ontology Language
P2P	Peer-to-Peer
PCIM	Policy Core information Model
PDDL	Plan Domain Description Language
PHP	Per-Hop Behaviour
PKI	Public Key infrastructure
RDF	Resource Description Framework
RIB	Routing information Base
QoS	Quality of Service
RMI	Remote Method Invocation
SLA	Service Level Agreement
SLA	Service Level Specification
SSL	Secure Socket Layer

TE	Traffic Engineering
UML	Unified Modelling Language
VPN	Virtual Private Network
WGS	World Geodetic System
XML	Extensible Markup Language

12 Acknowledgements

This deliverable was made possible due to the large and open help of the WP9 team of the EMANICS consortium within the Network of Excellence, which includes all of the deliverable authors as indicated in the document control. Many thanks to all of them.